

**AD-A258 424**



**AFIT/GSS/ENC/92D-1**

①

**DEVELOPMENT OF A STANDARD SET OF  
SOFTWARE INDICATORS  
FOR AERONAUTICAL SYSTEMS CENTER**

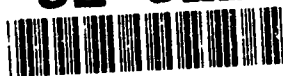
**THESIS**

**Bradley J. Ayres, B.S., M.A. Captain, USAF**  
**William M. Rock, B.S. Captain, USAF**

**AFIT/GSS/ENC/92D-1**

**DTIC**  
**SELECTE**  
**DEC 18 1992**  
**S B D**

012289 **92-32213**



412P8

**Approved for public release; distribution unlimited**

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

<b>Accession For</b>	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

DTIC QUALITY INSPECTED 2

**AFIT/GSS/ENC/92D-1**

**DEVELOPMENT OF A STANDARD SET OF  
SOFTWARE INDICATORS  
FOR AERONAUTICAL SYSTEMS CENTER**

**THESIS**

**Presented to the Faculty of the School of Systems and Logistics  
of the Air Force Institute of Technology  
Air University  
In Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science in Software Systems Management**

**Bradley J. Ayres, B.S., M.A.  
Captain, USAF**

**William M. Rock, B.S.  
Captain, USAF**

**September 1992**

**Approved for public release; distribution unlimited**

## Preface

The purpose of this research effort was to develop a standard set of software indicators for ASC and the procedures to implement them. These indicators will be used by software managers and engineers to improve management of software development efforts and by ASC/ENASC to provide better estimates for new projects.

In order to accomplish this research effort, a methodology was developed based on the Deming Management Method. This method is based on the concept of continuous process improvement. The set of indicators developed is therefore an initial first step in the process of providing managers and engineers at ASC the tools for better control of their software projects. Though the methodology used proved to be successful, it is certainly not the only method available, and we encourage others to try alternative methods.

This research would not have been successful without the help of the many managers and engineers at ASC that volunteered their valuable time to allow us to learn their insights into the software development process. We hope the results will be of benefit to them and wish them the best of luck in their future endeavors. We would also like to thank Professors Daniel Ferens and Daniel Reynolds for their invaluable assistance in the accomplishment of this research effort. And last, but certainly not least, we would like to thank our families for their patience and understanding through what was a most trying time for all of us.

Bradley J. Ayres

Bill "Bill" Rock



## Table of Contents

	Page
Preface.....	ii
List of Figures.....	x
List of Tables .....	xii
Abstract .....	xiii
1.0 Introduction .....	1-1
1.1 The Software Crisis.....	1-1
1.2 The Need for Improved Management of the Software Development Process.....	1-2
1.3 The Need for Software Indicators.....	1-3
1.4 Problem Statement .....	1-4
1.5 Research Objectives .....	1-4
1.6 Scope of Research .....	1-5
1.7 Definitions .....	1-6
1.8 Thesis Overview .....	1-6
2.0 Literature Review .....	2-1
2.1 Review of Software Metrics .....	2-1
2.1.1 How Indicators are Selected and Used.....	2-1
2.1.2 Review of Currently Available Indicators .....	2-7
2.2 The Deming Management Method.....	2-8
2.2.1 The Need to Improve Management of Software Development Efforts and the Deming Management Method .....	2-8
2.2.2 Some Fundamentals of the Deming Management Method .....	2-9
2.2.2.1 A Model of the Process.....	2-10
2.2.2.2 Operational Definitions.....	2-10
2.2.2.3 Special and Common Causes of Variance.....	2-12
2.2.2.4 The Deming Cycle .....	2-13
2.2.3 How the Deming Method Relates to This Research Effort.....	2-15
3.0 Methodology .....	3-1
3.1 Gathering Data About the Current Environment .....	3-2
3.1.1 Generating the Process Flow Diagram.....	3-3
3.1.1.1 Generating a General Flow Diagram .....	3-4
3.1.1.2 Revising the General Flow Diagram.....	3-5

	Page
3.1.1.3 Modifying the Revised Flow Diagram .....	3-5
3.1.1.4 Generating the Research Flow Diagram.....	3-6
3.1.2 Developing Research Questions From the Research Flow Diagram .....	3-8
3.1.2.1 Developing Research Questions for the Software Managers and Engineers Subprocess .....	3-9
3.1.2.2 Developing Research Questions for the ASC/ENASC Subprocess.....	3-15
3.1.3 Developing a Survey Instrument .....	3-19
3.1.3.1 Construction of Software Managers and Engineers Interviews .....	3-20
3.1.3.1.1 The General Information Section.....	3-20
3.1.3.1.2 The Job and Information Needs Section.....	3-21
3.1.3.1.2.1 Interview for Respondents Not Using Indicators .....	3-21
3.1.3.1.2.2 Interview for Respondents Using Indicators.....	3-21
3.1.3.1.3 Construction of the Data Sheets .....	3-22
3.1.3.1.3.1 The Software Areas Sheet.....	3-24
3.1.3.1.3.2 The Software Indicators Sheet .....	3-24
3.1.3.1.3.3 The Software Publications Sheet .....	3-25
3.1.3.1.4 The Final Interview Questionnaire.....	3-25
3.1.3.1.5 Validating the Interview Instrument.....	3-26
3.1.3.2 Construction of ASC/ENASC's Interview .....	3-27
3.1.4 Conducting the Interviews.....	3-27
3.1.4.1 Conducting the Software Managers and Engineers Interviews .....	3-28
3.1.4.1.1 Obtaining the Sample of Individuals to Interview .....	3-28
3.1.4.1.2 Scheduling the Personal Interviews .....	3-28
3.1.4.1.3 Changes to the Interview Questionnaire .....	3-29
3.1.4.2 Conducting the ASC/ENASC and Customer Interviews.....	3-31
3.1.5 Result of Defining the Current Environment .....	3-32
3.2 Documentation of the Current Environment.....	3-32
3.2.1 Question Response Analysis .....	3-33
3.2.1.1 Question Response Analysis Methodology .....	3-33
3.2.1.2 Question Response Analysis Outcomes.....	3-35
3.2.2 Analysis of Software Managers, Engineers, and ASC/ENASC Indicator and Area Preferences .....	3-35
3.2.2.1 Indicator Sheet Analysis Methodology.....	3-35
3.2.2.2 Area Sheet Analysis Methodology.....	3-37
3.2.2.3 Publication Sheets Analysis Methodology .....	3-37
3.2.2.4 Outcome of the Indicator and Area Preference Analysis.....	3-38
3.2.3 Analysis for the Impact of Variables.....	3-38
3.2.3.1 Impact of Variables Methodology.....	3-38
3.2.3.2 Outcome of Analysis of Impact of Variables .....	3-39
3.2.4 Analysis of the Research Flow Diagram.....	3-40
3.2.4.1 Flow Diagram Analysis Methodology.....	3-40

	Page
3.2.4.2 Flow Diagram Analysis Outcome.....	3-40
3.2.5 Outcome of the Documentation of the Current Environment.....	3-40
3.3 Conceptualization of the Improved Software Indicator Environment.....	3-41
3.3.1 Selection of the Standard Set of Indicators.....	3-42
3.3.1.1 Methodology for the Selection of the Standard Set .....	3-42
3.3.1.2 Development of Procedures for the Use of the Selected Indicators.....	3-43
3.3.2 Analysis for the Discovery of Process Problems.....	3-43
3.3.2.1 Creation of an Ideal Software Indicator Environment.....	3-44
3.3.2.2 Deriving the Standard Responses to the Survey Questions .....	3-47
3.3.2.3 Data Analysis Methodology for ASC/ENASC.....	3-49
3.3.2.4 Deriving Standard Responses to the ASC/ENASC Survey Questions.....	3-51
3.3.2.5 Outcome of Creation of the "Ideal" Software Indicator Environment.....	3-52
3.3.2.6 Comparison of Actual Responses to Ideal Responses for the Process Flow Diagram .....	3-52
3.3.2.7 Outcome of the Analysis of the Current Process.....	3-53
3.3.3 Comparison of Process Problems to Program Problems.....	3-53
3.3.3.1 Comparison Methodology.....	3-53
3.3.3.2 Comparison Outcome .....	3-53
3.3.4 Developing Solutions to Problems.....	3-53
3.3.5 Outcome of the Analysis.....	3-54
3.4 Implementation of the Improved Software Indicator Environment.....	3-54
4.0 Data Analysis .....	4-1
4.1 Analysis of Question Responses.....	4-1
4.1.1 Analysis of Questions Responses for Software Managers and Engineers.....	4-1
4.1.1.1 With Indicators Questionnaire .....	4-1
4.1.1.2 Without Indicators Questionnaire .....	4-9
4.1.1.3 Overall Findings.....	4-13
4.1.2 Analysis of Questions Responses for ASC/ENASC.....	4-14
4.2 Analysis of Indicators, Areas, and Publications Sheets .....	4-17
4.2.1 Results of Indicator Sheets Analysis.....	4-17
4.2.2 Results of Area Sheets Analysis .....	4-18
4.2.3 Results of Publication Sheet Analysis.....	4-19
4.2.4 Overall Analysis of Indicator, Area, and Publication Sheets.....	4-20
4.3 Analysis of Impact of Variables .....	4-24
4.3.1 Impact of Variables on the Use of Indicators.....	4-24
4.3.2 Impact of Variables on Needs.....	4-25

	Page
4.4 Analysis of the Research Flow Diagram.....	4-27
4.5 Selection of Standard Indicators.....	4-28
4.5.1 Selection of Schedule Indicator.....	4-36
4.5.2 Selection of a Cost Indicator.....	4-37
4.5.2.1 Selection of a Size Indicator.....	4-37
4.5.3 Selection of a Requirements Indicator.....	4-38
4.5.4 Selection of Software Performance Indicators.....	4-38
4.6 The Standard Set of Indicators With Recommendations for Use.....	4-38
4.6.1 Schedule Indicators.....	4-39
4.6.1.1 Resource Allocation Status.....	4-39
4.6.1.1.2 Frequency of Update.....	4-39
4.6.1.1.3 Update Criteria.....	4-39
4.6.3.1.4 Indicator Inputs.....	4-39
4.6.3.1.5 An Example of the Indicator.....	4-40
4.6.3.1.6 Using the Indicator.....	4-40
4.6.3.1.7 Tailoring Suggestions.....	4-40
4.6.1.2 Preliminary Design Status.....	4-40
4.6.1.2.1 Frequency of Update.....	4-42
4.6.1.2.2 Update Criteria.....	4-42
4.6.1.2.3 Indicator Inputs.....	4-42
4.6.1.2.4 An Example of the Indicator.....	4-42
4.6.1.2.5 Using the Indicator.....	4-44
4.6.1.2.6 Tailoring Suggestions.....	4-44
4.6.1.3 Detailed Design Status.....	4-44
4.6.1.3.1 Frequency of Update.....	4-44
4.6.1.3.2 Update Criteria.....	4-44
4.6.1.3.3 Indicator Inputs.....	4-45
4.6.1.3.4 An Example of the Indicator.....	4-45
4.6.1.3.5 Using the Indicator.....	4-45
4.6.1.3.6 Tailoring Suggestions.....	4-47
4.6.1.4 Code and Unit Test Status.....	4-47
4.6.1.4.1 Frequency of Update.....	4-47
4.6.1.4.2 Update Criteria.....	4-47
4.6.1.4.3 Indicator Inputs.....	4-47
4.6.1.4.4 An Example of the Indicator.....	4-48
4.6.1.4.5 Using the Indicator.....	4-48
4.6.1.4.6 Tailoring Suggestions.....	4-50
4.6.1.5 Integration Status.....	4-50
4.6.1.5.1 Frequency of Update.....	4-50
4.6.1.5.2 Update Criteria.....	4-50
4.6.1.5.3 Indicator Inputs.....	4-50
4.6.1.5.4 An Example of the Indicator.....	4-51
4.6.1.5.5 Using the Indicator.....	4-51
4.6.1.5.6 Tailoring Suggestions.....	4-51

	Page
4.6.2 Cost .....	4-53
4.6.2.1 Man Months of Effort.....	4-53
4.6.2.2 Frequency of Update .....	4-53
4.6.2.3 Update Criteria .....	4-53
4.6.2.4 Indicator Inputs.....	4-53
4.6.2.1.5 An Example of the Indicator.....	4-54
4.6.2.1.6 Using the Indicator.....	4-54
4.6.2.1.7 Tailoring Suggestions .....	4-54
4.6.2.2 Software Size .....	4-56
4.6.2.2.1 Frequency of Update .....	4-56
4.6.2.2.2 Update Criteria .....	4-56
4.6.2.2.3 Indicator Inputs.....	4-56
4.6.2.2.4 An Example of the Indicator.....	4-57
4.6.2.2.5 Using the Indicator.....	4-57
4.6.2.2.6 Tailoring Suggestions .....	4-57
4.6.3 Requirements .....	4-59
4.6.3.1 Requirements Stability.....	4-59
4.6.3.1.1 Frequency of Update .....	4-59
4.6.3.1.2 Update Criteria .....	4-59
4.6.3.1.3 Indicator Inputs.....	4-59
4.6.3.1.4 An Example of the Indicator.....	4-60
4.6.3.1.5 Using the Indicator.....	4-60
4.6.3.1.6 Tailoring Suggestions .....	4-60
4.6.3.2 Design Stability .....	4-60
4.6.3.2.1 Frequency of Update .....	4-60
4.6.3.2.2 Update Criteria .....	4-60
4.6.3.2.3 Indicator Inputs.....	4-60
4.6.3.2.4 An Example of the Indicator.....	4-62
4.6.3.2.5 Using the Indicator.....	4-62
4.6.3.2.6 Tailoring Suggestions .....	4-62
4.6.4 Software Performance.....	4-62
4.6.4.1 I/O Bus Throughput Capability .....	4-64
4.6.4.1.1 Frequency of Update .....	4-64
4.6.4.1.2 Update Criteria .....	4-64
4.6.4.1.3 Indicator Inputs.....	4-64
4.6.4.1.4 An Example of the Indicator.....	4-64
4.6.4.1.5 Using the Indicator.....	4-64
4.6.4.1.6 Tailoring Suggestions .....	4-64
4.6.4.2 Processor Memory Utilization .....	4-66
4.6.4.2.1 Frequency of Update .....	4-66
4.6.4.2.2 Update Criteria .....	4-66
4.6.4.2.3 Indicator Inputs.....	4-66
4.6.4.2.4 An Example of the Indicator.....	4-66
4.6.4.2.5 Using the Indicator.....	4-66
4.6.4.2.6 Tailoring Suggestions .....	4-68

	Page
4.6.4.3 Processor Throughput Utilization.....	4-68
4.6.4.3.1 Frequency of Update .....	4-68
4.6.4.3.2 Update Criteria .....	4-68
4.6.4.3.3 Indicator Inputs.....	4-68
4.6.4.3.4 An Example of the Indicator.....	4-69
4.6.4.3.5 Using the Indicator .....	4-69
4.6.4.3.6 Tailoring Suggestions .....	4-69
4.7 Results of Analysis of Process Problems .....	4-69
4.7.1 Input Responses With Indicators .....	4-69
4.7.2 Transform Responses With Indicators .....	4-71
4.7.3 Output Responses With Indicators .....	4-71
4.7.4 Voice of the Customer Responses With Indicators.....	4-72
4.7.5 Input Responses Without Indicators .....	4-73
4.7.6 Transform Responses Without Indicators .....	4-74
4.7.7 Output Responses Without Indicators .....	4-74
4.7.8 Voice of the Customer Responses Without Indicators.....	4-74
4.7.9 Summary of Analysis .....	4-75
4.8 Comparison of Problems.....	4-76
4.9 Solution to Problems .....	4-76
4.9.1 Contractual Issues.....	4-76
4.9.2 Selection of the Wrong Indicator .....	4-77
4.9.3 Lack of Knowledge About Indicators .....	4-78
4.9.4 Other Problems With Indicators .....	4-79
4.9.5 Implementing the Proposed Solutions.....	4-80
4.10 Vision of the Improved Process.....	4-80
4.10.1 Opportunity for Improvement .....	4-80
4.10.2 Improving the Current Process.....	4-81
 5.0 Conclusions and Recommendations.....	 5-1
5.1 Overview of the Research .....	5-1
5.1.1 Objectives of the Research.....	5-1
5.1.2 Research Methodology .....	5-2
5.2 Detailed Results of the Research.....	5-5
5.2.1 Results of Objective 1 .....	5-5
5.2.2 Results of Objective 2 .....	5-5
5.2.3 Results of Objective 3 .....	5-7
5.3 Recommendations for ASC/ENASC.....	5-7
5.4 Recommendations for Future Research Efforts.....	5-9
 Appendix A Indicator Database.....	 A-1
Appendix B Preliminary Data Sheet.....	B-1

	Page
Appendix C Interview Questionnaires.....	C.1-1
Appendix C.1 With Indicators Questionnaire.....	C.1-1
Appendix C.2 Without Indicators Questionnaires .....	C.2-1
Appendix C.3 ASC/ENASC Questionnaire .....	C.3-1
Appendix C.4 Database Customer Questionnaire .....	C.4-1
Appendix D Survey Responses.....	D.1-1
Appendix D.1 With Indicators Survey Responses .....	D.1-1
Appendix D.2 Without Indicators Survey Responses .....	D.2-1
Appendix D.3 ASC/ENASC Survey Responses.....	D.3-1
Appendix D.4 Database Customer Survey Responses.....	D.4-1
Appendix E Results of Analysis of Interview Questions .....	E-1
Appendix F Results of Indicator and Area Sheets.....	F-1
Appendix G Software Indicator Handbook.....	G-1
Bibliography.....	BIB-1
Vita.....	V-1

## List of Figures

Figure	Page
2.1 Generic Process Model.....	2-11
2.2 The Deming Cycle.....	2-13
2.3 Operationalized Deming Cycle .....	2-16
2.4 Research Actions.....	2-18
2.5 Steps, Tasks, and Objectives .....	2-19
3.1 Research Tasks .....	3-1
3.2 Task Relationship to Objectives.....	3-2
3.3 Task Spans.....	3-2
3.4 Generate Flow Diagram and Gather Data .....	3-3
3.5 Analyze Data.....	3-3
3.6 Develop Vision of Improved Environment .....	3-4
3.7 General Flow Diagram .....	3-5
3.8 Revised Flow Diagram.....	3-6
3.9 Modified Revised Flow Diagram .....	3-7
3.10 Research Flow Diagram .....	3-8
3.11 Software Managers and Engineers Subprocess.....	3-9
3.12 ASC/ENASC Subprocess .....	3-16
3.13 "Missing" VOC Relationship.....	3-17
3.14 Indicator/Area/Publication.....	3-26
3.15 Defining the Current Environment .....	3-32
3.16 Data Analysis Process .....	3-33
3.17 Data Analysis Methodology.....	3-41
3.18 Improved Environment Methodology.....	3-42
3.19 Development of the Improved Process.....	3-54
4.1 Weighted Average Analysis of Impact of Variables.....	4-26
4.2 Majority Response Analysis of Impact of Variables.....	4-26



Figure	Page
4.3 Final Flow Diagram.....	4-28
4.4 Example of Requirements Allocation Status Indicator .....	4-41
4.5 Example of Preliminary Design Status Indicator .....	4-43
4.6 Example of Detailed Design Status Indicator .....	4-46
4.7 Example of Code and Unit Test Status Indicator .....	4-49
4.8 Example of Integration Status Indicator .....	4-52
4.9 Example of Man Months of Effort Indicator .....	4-55
4.10 Example of Software Size Indicator.....	4-58
4.11 Example of Requirements Stability Indicator .....	4-61
4.12 Example of Design Stability Indicator .....	4-63
4.13 Example of I/O Bus Throughput Capability Indicator.....	4-65
4.14 Example of Processor Memory Utilization Indicator.....	4-67
4.15 Example of Processor Throughput Utilization Indicator .....	4-70
4.16 Current Environment .....	4-81
4.17 Vision of the Improved Environment.....	4-83
5.1 Research Objectives.....	5-3
5.2 Actions for Objective 1 .....	5-3
5.3 Actions to Begin Objective 2 .....	5-4
5.4 Actions to Complete Objective 2 .....	5-4
5.5 Indicator Process at ASC.....	5-5
5.6 Core Areas.....	5-6

## List of Tables

Table	Page
2.1 Deming Cycle Steps .....	2-14
3.1 Survey Questions .....	3-10
3.2 Informational Survey Questions.....	3-14
3.3 ASC/ENASC Survey Questions .....	3-18
3.4 ASC/ENASC Background Questions.....	3-19
3.5 Pareto Analysis Categories .....	3-36
3.6 Program Variables.....	3-39
3.7 Indicator Selection Criteria.....	3-43
3.8 Standard Inputs to Software Managers and Engineers.....	3-46
3.9 Ideal Survey Responses (Managers and Engineers) .....	3-47
3.10 Ideal Survey Responses for ASC/ENASC.....	3-51
4.1 Comparison of Responses to Actual (With Indicators).....	4-2
4.2 Comparison of Responses to Actual (Without Indicators).....	4-10
4.3 Comparison of Responses to Actual (ASC/ENASC) .....	4-15
4.4 Indicator Analysis for Managers' Responses.....	4-17
4.5 Indicator Analysis for Engineers' Responses .....	4-18
4.6 Area Analysis for Managers and Engineers.....	4-19
4.7 Publication Familiarization .....	4-20
4.8 Indicator to Publication Familiarity Cross-Reference .....	4-21
4.9 Core Areas From Area Sheets Analysis.....	4-22
4.10 Variables for Programs Not Using Indicators .....	4-24
4.11 Indicator Sources and Comparison.....	4-29
5.1 Research Objectives.....	5-1
5.2 Selected Indicators .....	5-6

Abstract

This research effort was directed to the development of a standard set of software indicators to be used by ASC. The research was sponsored by ASC/ENASC, who indicated their desire for such a standard set of indicators to allow a database to be developed of program data that would be useful in making predictions about future software efforts. The research began with an attempt to characterize the software indicator environment at ASC, and also to perform a literature search for software indicators currently available. The desired result was an initial core set of indicators useful to software managers and engineers and to ENASC and also a methodology for the continuous refinement of the set of indicators as part of a process to better define the software indicator environment. Thirty-four interviews were done on software managers and engineers and the results showed some common areas of interest and some unique interests for managers and some for engineers.

# DEVELOPMENT OF A STANDARD SET OF SOFTWARE INDICATORS FOR AERONAUTICAL SYSTEMS CENTER

## 1.0 Introduction

The focus of this research effort was to satisfy the need for a standard set of software indicators to be implemented within Aeronautical Systems Center (ASC). Software indicators are tools to help managers and engineers improve software development efforts. The need to improve software development efforts has arisen because of increasing schedule slips and cost overruns. These problems can be attributed to events that began in the early 1960s and resulted in a condition called "the software crisis."

### 1.1 The Software Crisis.

In the early 1960s, computer technology was dominated by hardware. Computer hardware was extremely expensive, and subsequent management focus naturally turned to controlling it (15:1-1). On the other hand, software development costs accounted for a minimal amount of the overall computer system costs. Therefore, software did not receive the management attention that computer hardware received (15:1-1).

During the 1960s, with the advent of the digital computer, microelectronics technology began to rapidly mature resulting in drastically reduced hardware costs and a greater demand for computer systems (15:1-1, 24:v). This increased demand for computer systems naturally resulted in an increased demand for software applications.

However, because of inadequate quality control and a lack of management attention in the early 1960s, many organizations were faced with poor management control and flawed technical development of software (15:1-1). This has led to software development not being able to keep pace with hardware and responding to the increasing demand for computer applications (9:1). This situation still exists today, and has led to a "software crisis".

The software crisis pertains to problems associated with the software development process (27:13-14). These problems are associated with developing software that will meet user requirements, the maintenance and operation of existing software, and the ability to keep pace with future requirements (7:3). The results of the software crisis have been increasing development costs, longer time periods to deliver end products, and an inability to satisfy customer requirements (27:13-14). In essence, the software development process is out of control, and quality software is not being produced.

#### 1.2 The Need for Improved Management of the Software Development Process.

The software crisis has had a significant impact on the Air Force. Software size estimates and developmental schedules are increasing at a seemingly uncontrollable rate. For example, the Peace Shield command and control program originally was estimated to be 100,000 lines of software code to be developed in 36 months; however, that estimate grew to 800,000 lines of code and the schedule slipped from 36 to 96 months (23:62). The end result was a 700 percent increase in the size of the software and a 167 percent increase in the schedule. This is just one of many Air Force programs facing a software crisis.

This situation has not gone unnoticed by Congress. In 1989, the House Appropriations Defense Subcommittee threatened zero funding for the Air Force's Advanced Tactical Fighter in order to force the Department of Defense (DOD) to focus on the software crisis (23:63).

ASC develops millions of lines of software code for its systems. The Advanced Tactical Fighter, mentioned previously, is one of the systems that ASC is currently developing. As this example indicates, the software crisis has had a direct impact on ASC. With declining budgets and an increasing demand for software, ASC must gain control of its software development processes. ASC needs to implement changes to improve management of the software development process and the quality of the software it produces.

### 1.3 The Need for Software Indicators.

The first step to improve management of a process is to provide data to allow visibility into what is happening within that process. As Lord Kelvin stated:

When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind... (27:89)

Likewise, William Sherkenbach states the following concerning

Dr. W. Edwards Deming:

Dr. Deming has said many times that management's job is prediction. This is because prediction increases knowledge, and knowledge is a prerequisite for action. When the predictions and the actions coincide, this increases wisdom. In order to predict, you must have data. (28:196)

Software indicators, or metrics, provide such data for the software development process. This is emphasized by DOD directing in DODD 5000.2, Defense Acquisition Program Procedures, that each service implement software indicators for its acquisition efforts (13:D-6-2).

Air Force Systems Command (AFSC) has published pamphlets AFSCP 800-43, Software Management Indicators, and AFSCP 800-14, Software Quality Indicators, that describe software indicators that may be used by organizations involved with the acquisition of software (2:1, 3:1). However, these pamphlets are only guides and the actual selection and implementation of software indicators is up to each software manager and engineer.

As part of their mission, the Computer Resources Branch of the Directorate for Avionics Engineering, ASC/ENASC, provides system engineering support and technical direction to the program offices within ASC in the areas of computer resources acquisition policy and standardization. To help them accomplish their mission, ENASC identified a need for the development of a standard set of software indicators for use by ASC software managers and engineers (5:1). A standard set of indicators not only supports the mission of ENASC, but it also helps software managers and engineers fulfill the requirements of DODD 5000.2. However, a standard set of software indicators by themselves would be incomplete. Only when they are combined with a set of procedures that fully define how to implement and utilize them within the current software indicator environment within ASC can their full effectiveness be realized.

#### 1.4 Problem Statement.

The purpose of this research is to develop a standard set of procedures and software indicators to be implemented by software managers and engineers within ASC's System Program Offices and incorporated by ASC/ENASC into a software program database.

#### 1.5 Research Objectives.

In order to fulfill the purpose of this research, the following objectives must be met:

1. Identify what data software managers and engineers need to improve software

development efforts and what ASC/ENASC needs to populate their database. Because software managers and engineers will have to use the standard set of software indicators developed by this research effort, their needs must be considered along with those of ASC/ENASC. This objective focuses on the software indicator environment within ASC to answer the question, "What data do software managers, engineers, and ASC/ENASC need?"

2. Identify the software indicators necessary to provide the data identified by the first objective. This objective focuses on software measurement techniques to answer the question, "How will the data needed by software managers, engineers, and ENASC be measured?"

3. Identify the procedures necessary to implement the software indicators identified by the second objective. This objective again focuses on the software indicator environment within ASC to answer the question, "What changes are required to implement the proposed software indicators?"

#### 1.6 Scope of Research.

The scope of this research will be limited to software development within ASC. The procedures and software indicators will be tailored specifically for software management and engineering within ASC. That is not to say that the methodology used will not be applicable to other organizations. However, no attempt will be made to generalize the methodology to other software development efforts outside the scope of ASC.

Due to time constraints and resource limitations, the research effort is also limited to those local activities within ASC and does not include activities of software developers outside of ASC.

It is beyond the scope of this research to develop and validate new software indicators if existing software indicators cannot obtain the data needed by ASC software managers, engineers, and ENASC. If a new software indicator is required, recommendations will be



provided as to what it should be, but validating it will not be accomplished by this research effort.

It is also beyond the scope of this research to test the proposed procedures and software indicators to obtain statistical evidence that they do improve the software management process. To do this would require control of software management processes within ASC which is not possible. In addition, time constraints would prohibit accomplishing such a test.

### 1.7 Definitions.

**Software Indicator:** A quantitative assessment of the degree to which a software product or process possesses a specified attribute (19:14). It may be a direct measure of a quantity associated with the software or an indirect measure based on primitives. For our purposes, software indicator and software metric will be synonymous.

**Primitive:** Data relating to the product or process of software development that is used in developing measures or quantitative descriptions of software. Primitives are directly measured or counted. (19:14)

**Software Management Process:** The process by which the development of software for delivery with a system is measured and monitored. It includes management of all phases of the software development effort to include requirements analysis, design, coding, testing, integration into an operational environment, and maintenance. It also includes evaluating the quality of the product and validating it against contract requirements. (22:4-6)

### 1.8 Thesis Overview.

The second chapter of this thesis is a literature review. It is divided into two parts. The first part is a literature review of different approaches to selecting software indicators and what indicators are currently available. Information from this review provided greater knowledge of

software indicators and their uses. However, it did not provide a suitable methodology for selecting a standard set of software indicators. As a result, literature written about the Deming management method was also reviewed. The second part of Chapter 2 provides the results of that review. It discusses how the Deming management method relates to the need for improving management of the software development process, what some of the fundamental principles of the Deming management method are, and how Deming's method applies to this research effort. At the end of Chapter 2, a set of tasks based on the Deming management method were identified and used as the basis for the research methodology.

The third chapter describes in detail the methodology used to satisfy the research objectives. Chapter 3 further expands and defines the tasks identified at the end of Chapter 2. It discusses what had to be accomplished, how the research was conducted, and the methodology used for data analysis.

The fourth chapter documents the results obtained from following the methodology described in Chapter 3. It describes the proposed standard set of software indicators for ASC and the procedures necessary to implement them.

The fifth chapter summarizes what was presented in the first four chapters, provides recommendations to ASC/ENASC for continuing a software indicator effort within ASC, and suggests areas for future research.

## 2.0 Literature Review

This literature review is divided into two parts. The first part reviews different approaches to selecting software indicators and which indicators are currently available to support the research effort. However, this review did not provide a suitable approach for selecting a standard set of software indicators. As a result, literature written about the Deming management method was also reviewed. The results of that review are contained in the second part of this chapter. The second part discusses how the Deming management method relates to the need for improving management of the software development process, what some of the fundamental principles of the Deming management method are, and how Deming's method applies to this research effort.

### 2.1 Review of Software Metrics.

This section of the literature review focuses on the actual software indicators themselves: how they are selected, how they can be used, and which ones are currently available.

2.1.1 How Indicators are Selected and Used. The literature recommends several methods for choosing indicators to establish an initial set. One approach was developed by Robert Grady and Deborah Caswell for Hewlett Packard. Their first task was to create a set of metrics for the company, but instead of focusing in on an existing model for the software process and its associated metrics, the decision was made to gather additional information with an eye toward creating a model unique to Hewlett Packard. They formed a group to guide the development and implementation of metrics called the "Software Metrics Council" (17:44-46). The metrics chosen by the Council corresponded to three "high level" Hewlett Packard goals: productivity, quality, and predictability

(17:46). They took the approach to sample more areas than they thought were likely to be needed to ensure that adequate data was taken.

This example shows one way of selecting indicators to start a program. First, a group is formed to select indicators based on "high level" organizational goals. Then, the group selects more indicators than are believed necessary with the idea of later reevaluation and eventual refinement of the set required. This means that a cycle of indicator selection and use will be established. An initial set of indicators will be selected, and the groundwork will have been laid for the next cycle which is a study of the effectiveness of the indicators selected to develop even more suitable sets of indicators in the future (17:44-46).

Though this approach worked well for Hewlett Packard, there are significant problems with applying it to this research effort. The primary one is the selection of members to make up the equivalence of a Software Metrics Council. This research effort has no control over ASC resources, and therefore cannot direct the formation of such a group to dedicate efforts to the selection of an initial set of metrics. In addition, the cost of implementing a set of indicators that represents more than is needed could be prohibitive.

Another approach to selecting indicators is to use the Goal/Question/Metric paradigm proposed by Victor Basili and David Weiss. The paradigm consists of six steps (6:729-732): 1) establish goals, 2) develop questions of interest from the goals, 3) establish data categories, 4) design and test a data collection form, 5) collect and validate data, and 6) analyze data. The data analysis should result in answers to the questions of interest which should determine whether or not the goals were achieved.

This approach was also determined to be unsuitable for the research effort. For this paradigm to be successful requires the establishment of data gathering goals and

questions of interest. This would require the formation of a group of software managers, engineers, and ASC/ENASC to work together to establish the goals and questions of interest. Just as in the case of the approach developed by Grady and Caswell for Hewlett Packard, this research effort cannot direct the formation of such a group to dedicate efforts to the establishment of the goals and questions of interest required for deriving the standard set of indicators.

Another source that suggested using the GQM paradigm to select the initial set of indicators is the Software Engineering Institute (SEI) at Carnegie-Mellon University. Even though their suggestion for selecting an initial set of indicators was determined to be unsuitable for the research effort, they provided a considerable amount of information concerning metrics and their uses. They believe ideal metrics have five characteristics. These characteristics are: simple and precisely definable, objective, easily obtainable, valid, and robust (29:4). They classify metrics into two categories: product and process (29:4). SEI draws a connection between the use of metrics and the use of models. The database of metric information can be used as a means of calibrating the models. However, they warn that this has had fairly good results when the programs in the database were all similar ( $\pm 20$  percent), but studies show that it does not work well for a new program that involves a different language, methodology, or environment from those in the database (29:13). They provide a list of several composite models. Composite models are models that combine intuition, statistical analysis, and expert judgement (29:12). The composite models listed include COCOMO and the Software Productivity, Quality, and Reliability Model (SPQR) (29:12).

The SPQR model was developed by Capers Jones. Jones states three purposes for having measurements: 1) to answer questions concerning software costs, 2) to determine how much time is need to deliver software, and 3) to understand why

significant variations occur (20:239). Jones says the understanding of why significant variations occur is the most important of the three because without it the knowledge gained from knowing costs and schedules is only frustrating (20:239). Jones also provides some cost information concerning using metrics. He states that IBM spends approximately 5 percent of their development costs on measurement-related activities, but typical programs should spend between 1.5 percent to 3 percent because they will only be collecting a subset of the amount of information that IBM collects (20:237).

Like SEI, Tom DeMarco also draws a connection between metrics and estimation. DeMarco provides additional characteristics of a good metric. These characteristics are: measurable, independent (cannot be influenced by project personnel), accountable (to ensure integrity), and precise (11:50). He also believes that there are two categories of metrics. However, his categories are different than those of the SEI. His categories are: result metrics and predictor metrics (11:54). Result metrics provide data about a completed system such as total cost or total manpower (11:54). Predictor metrics provide data that is used to forecast what the expected end value will be (11:54). Result metrics and predictor metrics are used in the process of cost projection (11:56). Result metrics are used to build an empirical database. They represent the numbers that are trying to be forecasted. Predictors are correlated to actual measurements in order to provide better and better estimations. DeMarco, like Grady and Caswell, also advocates the establishment of a Metrics Group that is solely responsible for the metrics program (11:36-37). He also provides data concerning the cost of having a metrics program. He states that the cost of establishing a metrics program is approximately 5 percent to 10 percent of the total manpower of the software development effort (11:36).

Roger Pressman also discusses the role of metrics and software data in determining cost estimates. Pressman states that accurate estimates are developed based on a historical baseline (28:95). The data used to establish this baseline should be collected in an ongoing manner (28:97). The data from the baseline is used to develop models, and the cost estimates from these models are only as good as the historical data the model is based on (28:101).

Though he does not prescribe a methodology, Samuel Conte does identify two major sources of data from which indicators can be selected. These sources are historical data and data from performing experiments (9:113). Conte also lists characteristics metrics should have. He identifies five key areas to use as a basis for evaluating the quality of an indicator: simplicity, validity, robustness, prescriptiveness, and analyzability (9:22). Conte also provides several items to be cautious of when using indicators. One of these is that care should be used in making comparisons of metrics that may not have the same meaning in different settings. Another is that calibration against similar programs is recommended before using any indicators in a new environment (9:27). Other factors to watch out for are the probabilistic nature of prediction metrics, and that the cost of a metric may outweigh its usefulness (9:28).

Tom Gilb provides eleven principles for motivating people to accept and use metrics. One of these principles is that metrics must be directly linked back to the needs of whomever is sponsoring the effort (16:397). Metrics should be direct measures of the objectives of the customer. This results in a greater chance for success by ensuring that the needs of the customer are being satisfied. His bottom line is that people have valid fears when it comes to implementing a metrics program, and you must be sensitive and sympathetic to that (16:401). The fear of metrics must be removed by creating an

environment where they are not justified any more (16:401). Following his eleven principles can help promote such an environment.

John Marciniak and Donald Reifer do not provide a specific methodology for selecting a set of software indicators, but they do provide advice on tailoring a set to meet their needs. They provide a basic set of eight indicators with caveats that straightforward, short duration, not overly complex programs may tailor the set to meet specific needs because the additional resources required to provide data for the full set of indicators may be prohibitive (22:139). They believe indicators provide the visibility into the software process that managers need to do their jobs effectively, and are important in giving the customer insight into the project so that the customer and the manager can work together to solve problems. In addition, caution is advised when using indicators to compare programs from different types of developments because it may lead to erroneous conclusions (22:168-9).

Watts Humphrey states that software data is gathered for the purpose of learning how to improve the process (18:331). He lists four objectives of data gathering: understanding, evaluation, control, and prediction (18:302). He also stresses that successful measurement is due to having a defined model to which measurements are compared (18:303). He also provides data on the cost of having a metrics program. Data from the Software Engineering Laboratory at the National Aeronautics and Space Administration showed that up to 15 percent of development costs were spent in gathering software data on a large number of measures across several programs (18:304). An interesting aspect of data analysis that Humphrey's discusses is the use of statistical control charts for analyzing software data (18:324). Humphrey's makes reference to Dr. W. Edwards Deming's work in the area of statistical process control (18:3).



Cho also refers to the work of Dr. Deming and of applying statistics to the development of software. Ten of Deming's fourteen points are addressed, giving specific details on how to apply these to software (8:14-15).

Though these sources did not provide a suitable methodology for selecting an initial set of software indicators, they did provide considerable information concerning the use of indicators. They provided characteristics that ideal indicators should have. They provided a connection between software indicators and models to make predictions, estimates, and improvements to the software process. The review also provided guidance on implementing a software indicator program within an organization. Finally, there were references to the works of Dr. Deming and the applicability to software development and data gathering. The Deming Management Method was reviewed to determine if it could provide a methodology for the selection of an initial set of indicators. That review is discussed in Section 2.2.

2.1.2 Review of Currently Available Indicators. This section of the literature review is contained in Appendix A. It is a compendium of over 65 software indicators drawn from 7 different sources, including Air Force Systems Command pamphlets 800-43 and 800-14, technical reports including Electronic Systems Division technical report ESD-TR-88-01, and books. The indicators are listed alphabetically and include information on source, primitives, the type of indicator (process or product), what areas of concern the indicator can be used for (cost, schedule, requirements, testing, etc.) and a short description of how the indicator is computed. A dashed line separates entries. If the same indicator is known by different names in different sources, it is listed under one name, and the alternative is given in the title or in the description.

## 2.2 The Deming Management Method.

Even though a considerable amount of information about software indicators was obtained in 2.1, Review of Software Metrics, an acceptable method for selecting a standard set of indicators could not be found. However, several sources of indicator information in the review did make reference to the work of Dr. W. Edwards Deming. As a result, a review of the Deming Management Method was conducted in order to ascertain if it could provide a methodology for selecting a standard set of software indicators. The following discussion shows how the Deming Management Method relates to the need for improving management of the software development process, what some of the fundamental principles of the Deming Management Method are, and how the Deming Management Method applies to this research effort.

2.2.1 The Need to Improve Management of Software Development Efforts and the Deming Management Method. In Chapter 1, it was stated that with declining budgets and an increasing demand for software, ASC must gain control of its software development processes. To endure the software crisis, the ASC needs to implement changes to improve management of the software development process and the quality of the software it produces. Also stated was the need to have software indicators that provide visibility into the management process.

At first, this situation would appear to be an excellent candidate for the DOD's Total Quality Management (TQM) effort. Among other things, TQM emphasizes the following ideals: a long-term commitment to continuous improvement, focus on process rather than product, rigorous analysis of management systems and processes, customer satisfaction (internal and external), and quality awareness throughout the organization (10:1.13b). But these ideals were not originated by the TQM effort. A primary source

that acted for the basis of the TQM effort and originated these ideals was the works of Dr. W. Edwards Deming (10:1.13b, 32:148-149).

It is Deming's management method that is primarily attributed with the tremendous success of the turnaround of the Japanese economy after World War II (31:6-21). Now American corporations are also adhering to the Deming management method in hope that it may also do for America's economy what it did for Japan's (26:20). As previously stated, even DOD has acknowledged the importance of Deming's management method through the implementation of their TQM effort.

However, many people believe that Deming's methods are applicable only to manufacturing or production processes. This is not true. Deming's method can also be applied to service organizations such as banks, restaurants, and software development (12:184). Deming's method is a universal approach to continuous process and quality improvement. As such, it can be applied to the ASC's problem of dealing with the software crisis, and developing software indicators to help improve management of the software development process.

2.2.2 Some Fundamentals of the Deming Management Method. Deming's management method is based on three components: an understanding of variance, an all-encompassing concept of quality, and the establishment of an environment from which knowledge and practice of the other two components can increase and continue (26:29). The third component can be accomplished by following Deming's Fourteen Points for management and avoiding his Deadly Diseases and Obstacles (12:18-148). TQM is DOD's implementation of the third component (10:1.13b). It is the other two components that are of primary interest to this research effort. To investigate these components further, the following aspects of the Deming method will be discussed: a model of the "process", operational definitions, special and common causes of variance,

and the Deming cycle. An understanding of these aspects will lead to an understanding of variance and the all-encompassing concept of quality.

2.2.2.1 A Model of the Process. An important aspect of the Deming Management Method is a shift in focus from concentrating on the end product to concentrating on the process which produces the end product. In its basic form, a process takes inputs and transforms them into outputs. A process can be imagined as being comprised of three blocks: inputs, transformation, and outputs. Inputs flow into transformation. Outputs flow from transformation to the customer. Within each process there are essentially two "voices" or perceptions of what is happening within the process. The first is the voice of the process itself. The voice of the process is represented by the actual output produced. The second is the voice of the customer. The voice of the customer is represented by what the customer needs. The two voices ideally should be the same; i.e., the process produces what the customer needs. But this is not always the case in the real world, and what is left is a "gap" between what the process produces and the needs of the customer. What management has to do is to continually strive to reduce the gap between the two voices. The difficulty lies in reducing variation and narrowing the gap. Both the voice of the process and the voice of the customer contain some variability. As variability is reduced, quality is improved. This model is illustrated in Figure 2.1. (28:7-19)

2.2.2.2 Operational Definitions. One way to begin to reduce the gap is through the use of operational definitions. According to Deming, an operational definition is "one that people can do business with" (12:277). What does he mean by this? He gives a good example of a blanket with a label that says "50 per cent wool." Does this mean that half the blanket is wool and the other half is some other material, or does it mean that the fibers of the blanket are half wool and half some other

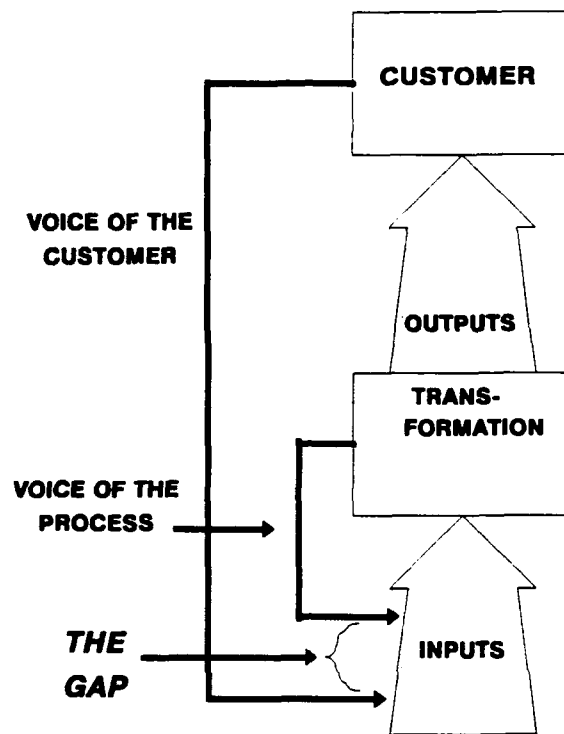


Figure 2.1 Generic Process Model

(28:11)

material? It is impossible to tell which one is true simply by the label, "50 per cent wool." (12:287-288, 28:211-212). The label is not of the form of an operational definition because it leaves itself open to much interpretation (26:110). An operational definition takes concepts and gives them communicable meaning (12:276). Operational definitions provide a linkage between the voice of the customer and the voice of the process (28:210). When a definition is truly operationalized, it then becomes understood by all parties involved. To do this, an operational definition must specify both the characteristics to be studied and how they will be measured in such a way that any competent individuals using this definition would all classify the characteristics the same (14:53). Using operational definitions reduces the gap between the voice of the

customer and the voice of the process which results in decreased variability and improved quality.

2.2.2.3 Special and Common Causes of Variance. Another way to reduce the gap is by understanding the causes of variability and trying to eliminate them. There are two kinds of variability in the output of the process: controlled variability due to the process itself and uncontrolled variability due sources outside the process (26:23). These kinds of variability are due to two things: common causes and special causes (26:23). Faults in the system are considered to be common causes of trouble, where as faults from passing events are considered to be special causes (12:314). By far the large majority of faults are due to common causes. Deming believes that 94 percent of faults are due to common causes and 6 percent are due to special causes (12:315). A general rule to follow is the 85-15 rule where 85 percent of what goes wrong is due to the system and only 15 percent of what goes wrong is due to individuals (32:20). How are common and special causes discovered? They are discovered through the use of control charts (12:310-312).

Control charts generally plot the output of the process over time or for different runs of the process. The control chart contains upper and lower control limits, and provide information about the state of the process. (32:24)

If the process is stable, it is in statistical control, and the variability of the process will not exceed the control limits. This variability is due to common causes. If a process is out of control, the variability will exceed the control limits. This variability is due to special causes. When a process is stable, all known sources of special causes have been eliminated. It is now the responsibility of management to further reduce the sources of common causes thus reducing the variability of the process. (12:321-322)

2.2.2.4 The Deming Cycle. To bring together and focus the activities to reduce variation requires the establishment of a framework or methodology. The Deming Cycle, or Wheel, provides such a framework. The Deming Cycle is comprised of four phases: plan, do, study, and act, as shown in Figure 2.2.

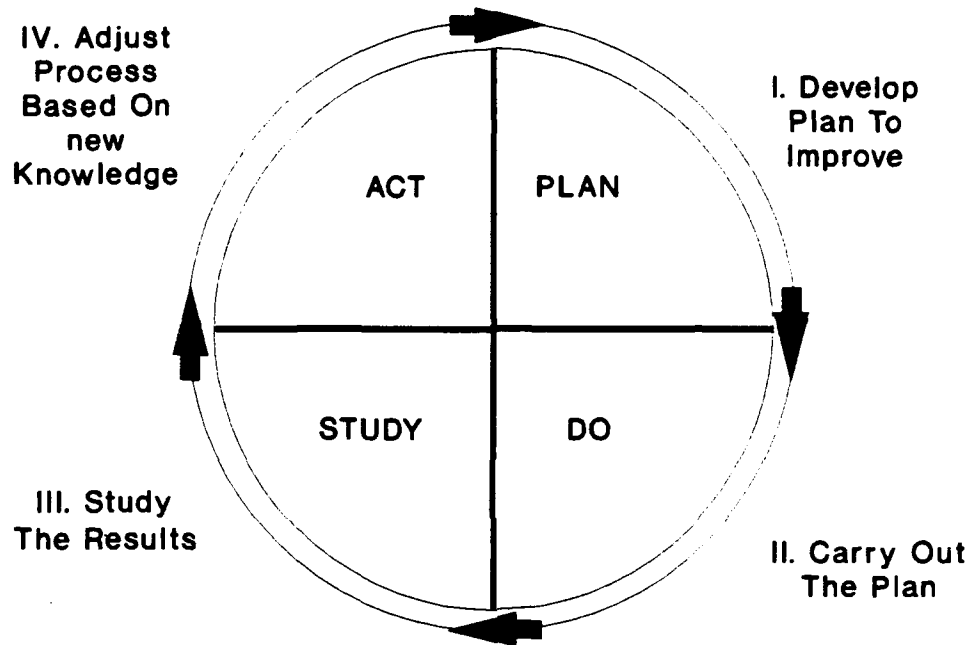


Figure 2.2 The Deming Cycle

(28:61)

In the planning phase, a plan is developed aimed at improvement. In the doing phase, the plan is then carried out. In the study phase, the results from carrying out the plan are studied. In the acting phase, adjustments are made based on new knowledge gained from the results of the study phase and the cycle is repeated (12:88, 28:63, 26:9).

To further define the Deming Cycle, eight action steps are created as shown in Table 2.1.

TABLE 2.1  
DEMING CYCLE STEPS

I. PLAN - DEVELOP PLAN TO IMPROVE

- Step 1: Identify the opportunity for improvement.
- Step 2: Document the present process.
- Step 3: Create a vision of the improved process.
- Step 4: Define the scope of the improvement effort.

II. DO - CARRY OUT THE PLAN

- Step 5: Implement plan on small scale over time with customers.

III. STUDY - STUDY THE RESULTS

- Step 6: Observe what you learned about the improved process improvement.

IV. ACT - ADJUST PROCESS BASED ON NEW KNOWLEDGE

- Step 7: Operationalize the new mix of resources.
- Step 8: Repeat the Cycle for the next opportunity.

(28:63)

In Step 1, the voice of the customer is compared to the voice of the process. When there is a "gap" between the two, an opportunity for improvement exists and is identified. (28:64)

In Step 2, once an area of improvement has been identified, it is then documented to characterize what the current process looks like. During this step, a process flow diagram or process map is generated and documented. By doing this, knowledge of the current process is obtained which leads to an understanding of the interdependencies, pitfalls, and inefficiencies within the process. (28:66)

In Step 3, a vision of the improved process is created. The knowledge of the current process that was obtained in Step 2 is analyzed to take into account any pitfalls or inefficiencies that were identified. Creating the vision of the improved process is aided by operationally defining the needs of the customer. The operational definition is



used to evaluate whether or not potential process improvements add value by closing the "gap" between the voice of the process and the voice of the customer. (28:68-69)

In Step 4, a plan is developed that describes the necessary actions for implementing the improved process. The plan must address the who, what, when, where, why, how and how much to implement the improved process. (28:70)

In Step 5, an experiment is conducted to obtain data on how the improvement actions affected the process. Experiments should be conducted on a small scale to minimize risk to the organization. (28:72)

In Step 6, the results from the experiment are analyzed with respect to its effectiveness at lessening the "gap" between the voice of the customer and the voice of the process. (28:74)

In Step 7, the improvement plan is implemented on a large scale throughout the organization. First, the plan is updated to include new knowledge gained from conducting the small scale experiment. Then the plan is put into operation by those managers who were determined to be essential to the improvement effort. (28:77)

In Step 8, the gaps between the voice of the customer and the process are again looked at to identify new opportunities for improvement and the cycle is then repeated. (28:78)

The operationalized Deming Cycle is shown in Figure 2.3.

### 2.3 How the Deming Method Relates to This Research Effort.

The eight Step Deming Cycle combined with the other fundamentals of the Deming Management Method provide a viable framework from which process control, along with continual improvement of the process and quality of the product can be achieved. An inherent part of this framework is the availability of data to determine the status of the process and impact of changes, and what happens to that data once it is

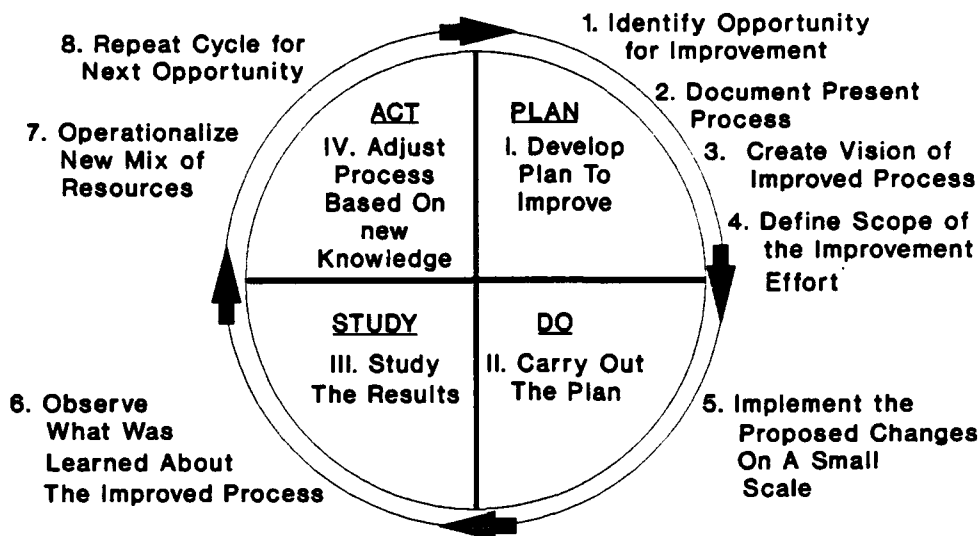


Figure 2.3 Operationalized Deming Cycle

obtained. Software indicators are critical to providing the data necessary to be able to determine the status of ASC software development efforts the and progress towards improved process control and software quality. In addition, the framework can also be used as part of this research effort to identify a methodology for identifying a standard set of software indicators for ASC and the procedures to implement them.

As stated in Chapter I, it is out of the scope of this research effort to test the proposed procedures and software indicators. This limits the research effort to the first part of the Deming Cycle, "Plan". However, the remaining parts of the cycle provide a necessary framework for continuing improvement and will be addressed in Chapter V as recommendations to ASC/ENASC. As stated previously, the planning portion of the Deming Cycle is further broken down into four steps: 1) identify opportunity for improvement, 2) document the present process, 3) create the vision of the improved process, and 4) define the scope of the improvement effort. These four steps will serve as the basis from which the research framework will be developed.

Step 1 of the Deming Cycle is to identify an opportunity for improvement. It is the starting point of the cycle. As previously stated in Chapter I, the software crisis has created a need for improved management. Implementing software indicators is the first step towards improving management, but a standard set of software indicators does not exist within ASC. ASC/ENASC, based on the need for a set of standard software indicators for ASC, requested that a research effort be initiated (5:1). In this instance, there exists an identified gap between the voice of the process, which does not have a standard set of indicators, and the voice of the customer, which requests that a standard set be developed. This identification of a need for a standard set of software indicators translates to the "opportunity for improvement" in Step 1 and becomes the starting point for this research effort. Therefore, Steps 2, 3, and 4 remain to be accomplished by this research effort.

Step 2 of the Deming Cycle is to document the present process. The "process" of interest to this research effort is the current software indicator environment within ASC. Therefore, Step 2 requires developing and documenting a flow diagram of the current software indicator environment within ASC.

Step 3 of the Deming Cycle is to create a vision of the improved process. If the current process is defined as the existing software indicator environment within ASC, then the "vision of the improved process" must be the existing software indicator environment improved by the implementation of the standard software indicators developed by this research effort. Therefore, Step 3 requires identifying the standard set of indicators for ASC by utilizing the knowledge gained in Step 2. Information necessary to document the current environment and the needs of the customer will come from software managers, engineers, and ASC/ENASC.

Step 4 of the Deming Cycle is to define the scope of the improvement effort. The scope of the improvement effort would be the actions necessary to implement the proposed standard set of software indicators for ASC. Therefore, Step 4 requires defining the procedures necessary to implement the proposed software indicators by recommending the who, what, when, where, why, how and how much is required to implement the standard set of software indicators.

The three steps defined above, Steps 2, 3, and 4, serve to establish an initial framework for the research effort and complete the first part of the Deming Cycle, "plan". In summary, the three steps are: 1) define the current software indicator environment within DOD; 2) define the improved environment based on standard software indicators; 3) define the actions necessary to implement the standard software indicators.

To accomplish these steps, the tasks shown in Figure 2.4 must be accomplished as described above.

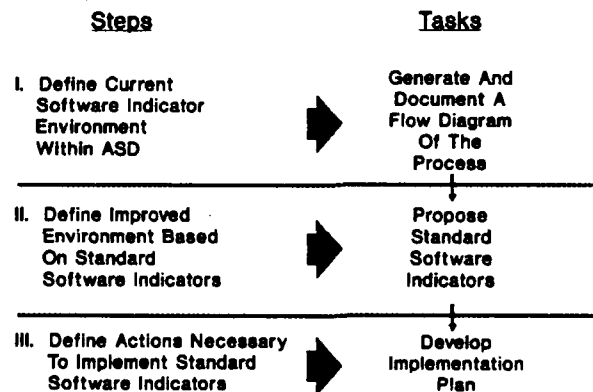


Figure 2.4 Research Actions

By accomplishing these steps and tasks, the research objectives in Chapter I. will be satisfied. The relationship between the steps, tasks, and research objectives are as shown in Figure 2.5.

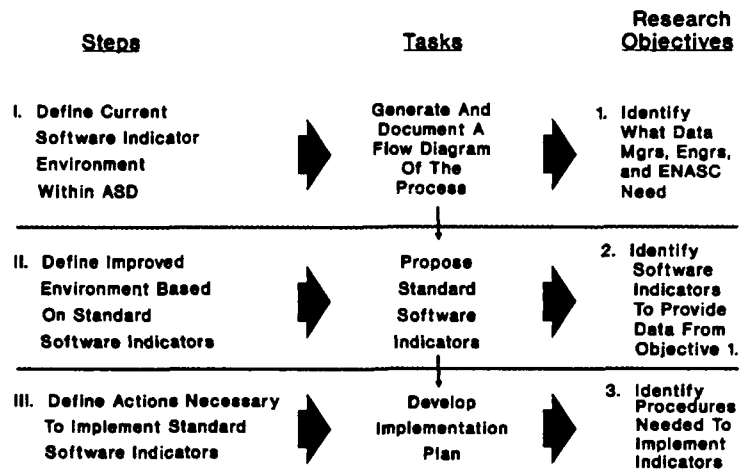


Figure 2.5 Steps, Tasks, and Objectives

Chapter 3 will further expand on the steps and tasks above to develop a methodology to address what needs to be done to satisfy the research objectives.

### 3.0 Methodology

Chapter 2 concluded with relating the research objectives to a set of tasks as shown in Figure 3.1. Acting on these tasks requires additional details about the tasks identified.

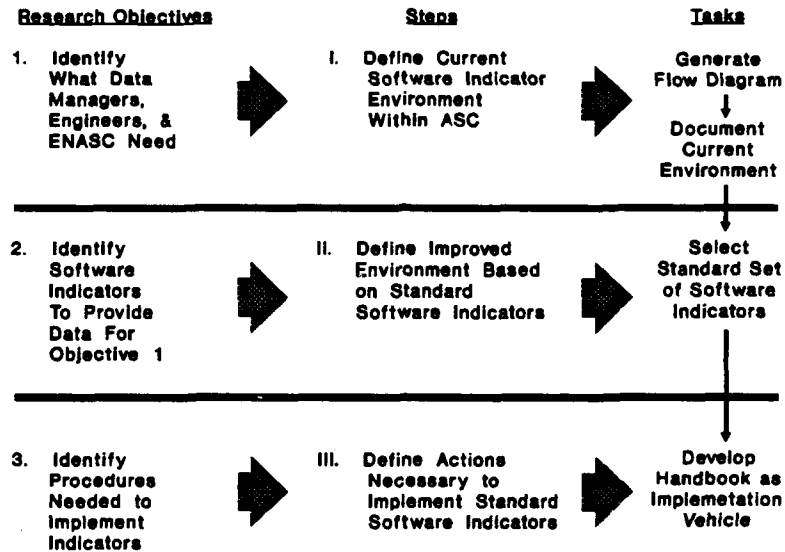


Figure 3.1 Research Actions

The tasks were operationalized into four subtasks that are the basis for the methodology.

The relationship of the research objectives to the subtasks is shown in Figure 3.2. The inputs and outputs of each task are shown in Figure 3.3.

Decomposition of these tasks into individual actions to be accomplished is shown in Figures 3.4 to 3.6.

This chapter will describe how the flow diagram was generated, how the interview questionnaire which was used to gather data to document the flow diagram was developed, how the interviews were conducted, and the methodology for analyzing the data collected from the interviews.

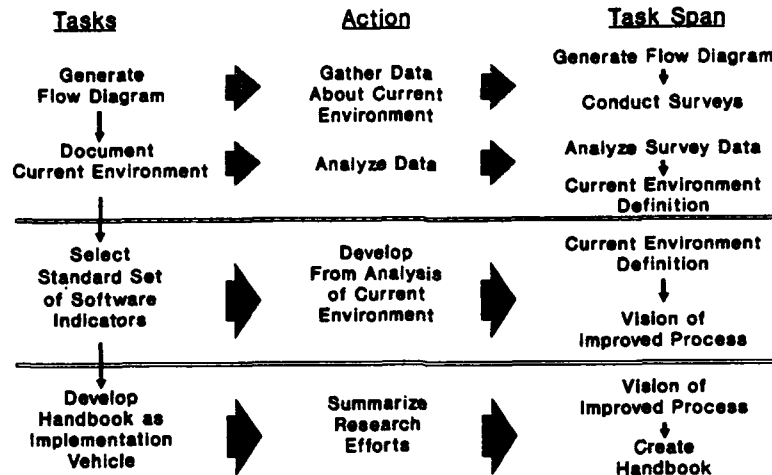


Figure 3.2 Task Relationship to Objectives

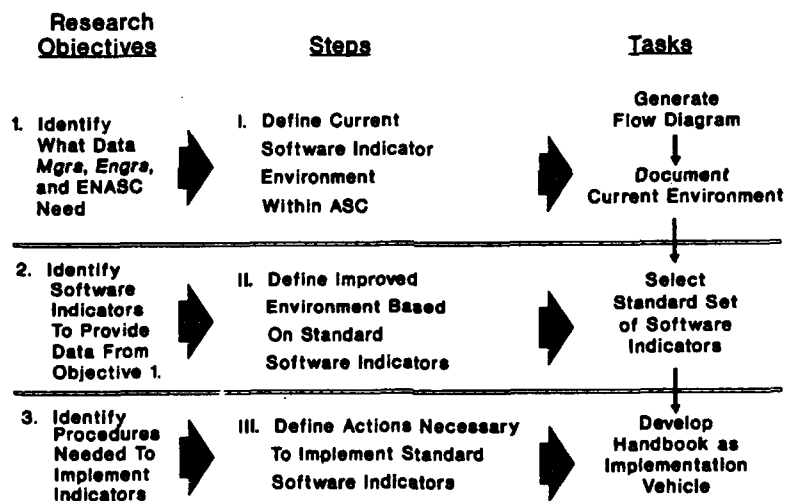


Figure 3.3 Task Spans

### 3.1 Gathering Data About the Current Environment.

The steps to gathering the data for the research effort are shown in Figure 3.4. The first two steps, identify the need for improvement and develop attack methodology, have already been accomplished in the first two chapters of the thesis. The first step in gathering the data is therefore to generate the process flow diagram.

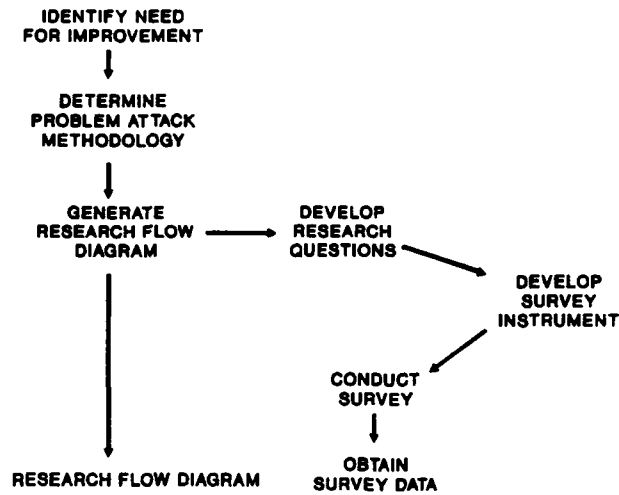


Figure 3.4 Generate Flow Diagram and Gather Data

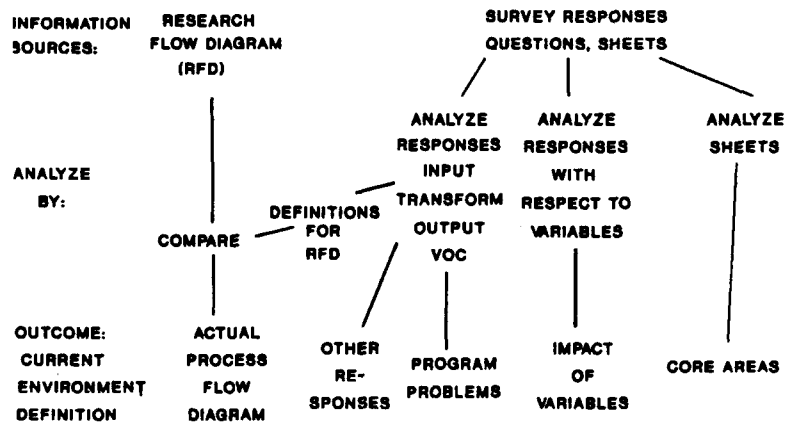


Figure 3.5 Analyze Data

3.1.1 Generating the Process Flow Diagram. One of the prime aspects of the Deming Management Method is a shift from focusing on the product to focusing on the process. Therefore, it makes sense that the research tasks should start with the development of a process flow diagram. As previously stated, a process can be imagined as being comprised of three basic blocks: inputs, transformation, and outputs. There are two perceptions of what is happening within the process: the voice of the process



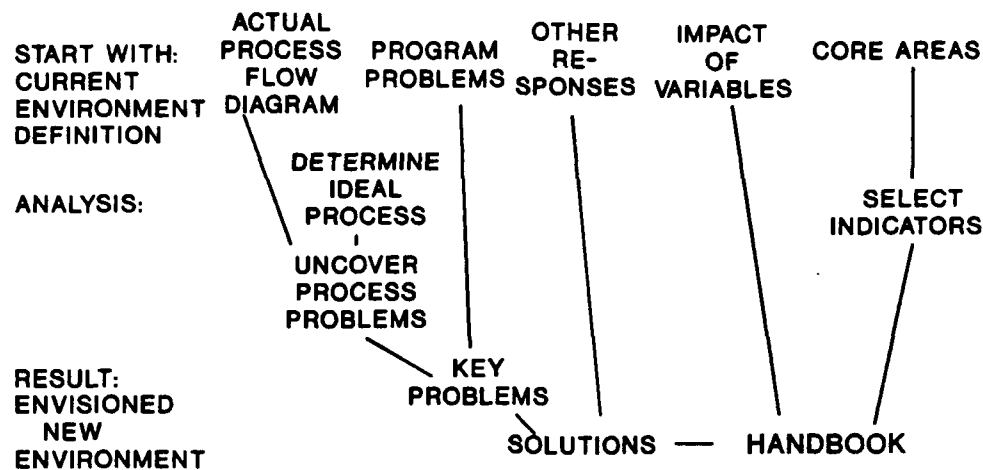


Figure 3.6 Develop Vision of Improved Environment

represented by what is actually produced, and the voice of the customer represented by what the customer actually wants or needs. In this task, these elements of a process were utilized in the generation of a process flow diagram for the software indicator environment within ASC.

3.1.1.1 Generating a General Flow Diagram. To begin the task of generating a process flow diagram, efforts were concentrated on developing the three blocks: inputs, transformation, and outputs. In the beginning of the research effort, it was believed that software indicator data would be inputs to the software developer who would transform the data into outputs to the ASC software managers and engineers. Then, the managers and engineers would transform the data into outputs to ASC/ENASC. It was believed that ENASC then may transform the data to produce a data report as an output to higher-level management. The Commander of ASC, ASC/CC, was used to represent higher-level management. The combination of this information resulted in the generation of a general flow diagram shown in Figure 3.7.

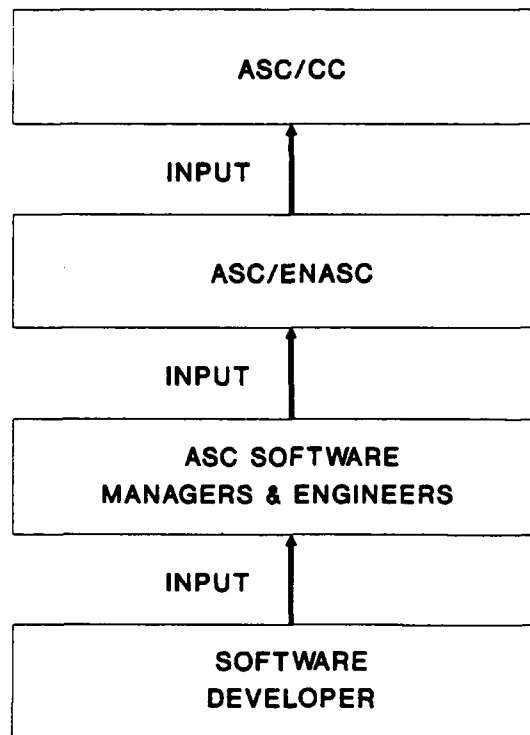


Figure 3.7 General Flow Diagram

3.1.1.2 Revising the General Flow Diagram. The general flow diagram was then reviewed by the research sponsor, ASC/ENASC. Based on comments made by ENASC, the general process flow diagram was revised. The revision included the deletion of ENASC providing higher-level management a report and the inclusion of the software indicators as inputs at the lowest level of the process flow diagram. The revised flow diagram is shown in Figure 3.8.

3.1.1.3 Modifying the Revised Flow Diagram. After reviewing the revised flow diagram, it was decided that there were two major contributors to the software indicator inputs that should be incorporated into the flow diagram. These two contributors were the currently available indicators as listed in Appendix A., and the indicators that were currently being used by ASC software managers and engineers. It was believed that the

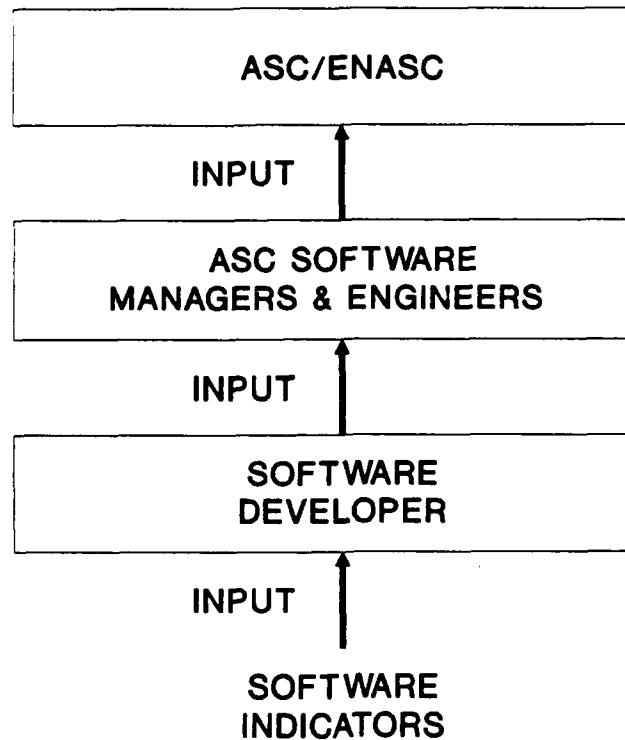


Figure 3.8 Revised Flow Diagram

standard set of software indicators for ASC would be influenced by these two contributors, and as such they should be included in the flow diagram. The revised flow diagram as was modified with the resulting flow diagram shown in Figure 3.9.

3.1.1.4 Generating the Research Flow Diagram. This provided the initial three blocks of the flow diagram from which the current environment could be documented. This information would form the basis for the "voice of the process" or what the process was actually producing. The next step was to incorporate the aspect of the "voice of the customer" into the process flow diagram.

It was envisioned that there would be three "customers" within the software indicator environment: the ASC software managers, the ASC software engineers, and ASC/ENASC. The ASC software managers and engineers would use software indicators

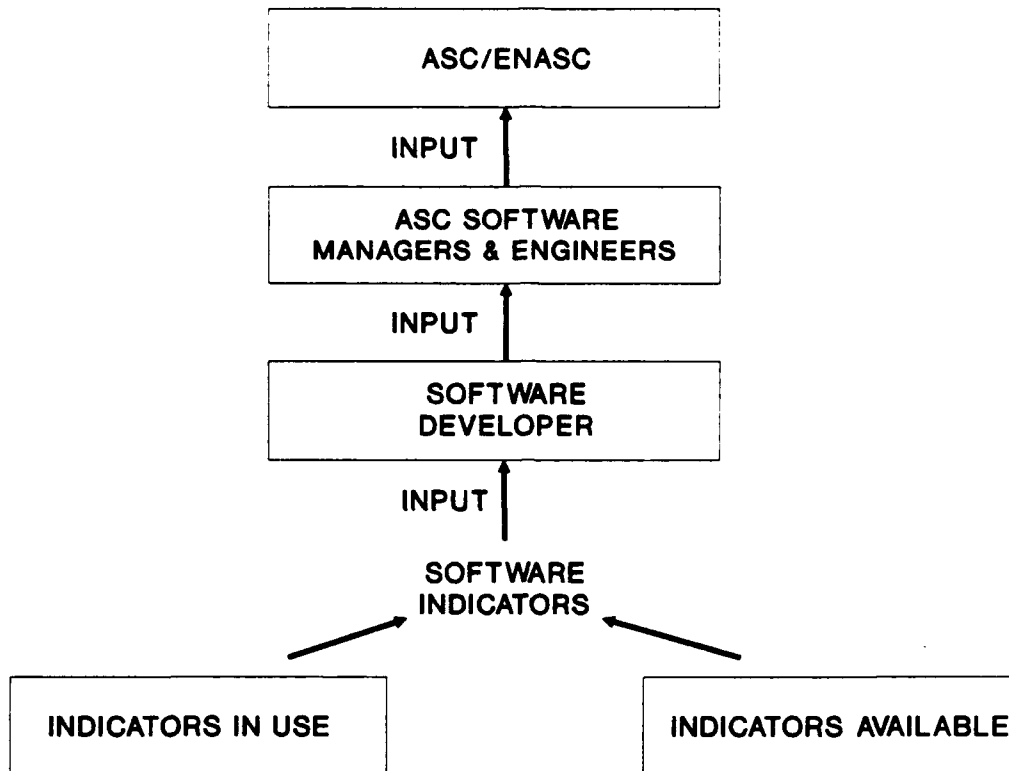


Figure 3.9 Modified Revised Flow Diagram

to improve management for each of their programs, whereas ASC/ENASC would utilize software indicators to improve management of software development efforts across all of ASC. The new flow diagram, which is referred to as the research flow diagram is shown in Figure 3.10.

This research flow diagram was reviewed by ASC/ENASC, the thesis committee, and ten of the computer resource focal points within ASC. No recommendations to change the research flow diagram resulted from this review.

The research flow diagram illustrates how continual improvement can be accomplished. By comparing the needs of the software managers, engineers, and ASC/ENASC to the actual products that are produced, the software indicator inputs can be modified, narrowing the gap between the voice of the customer and the voice of the process.

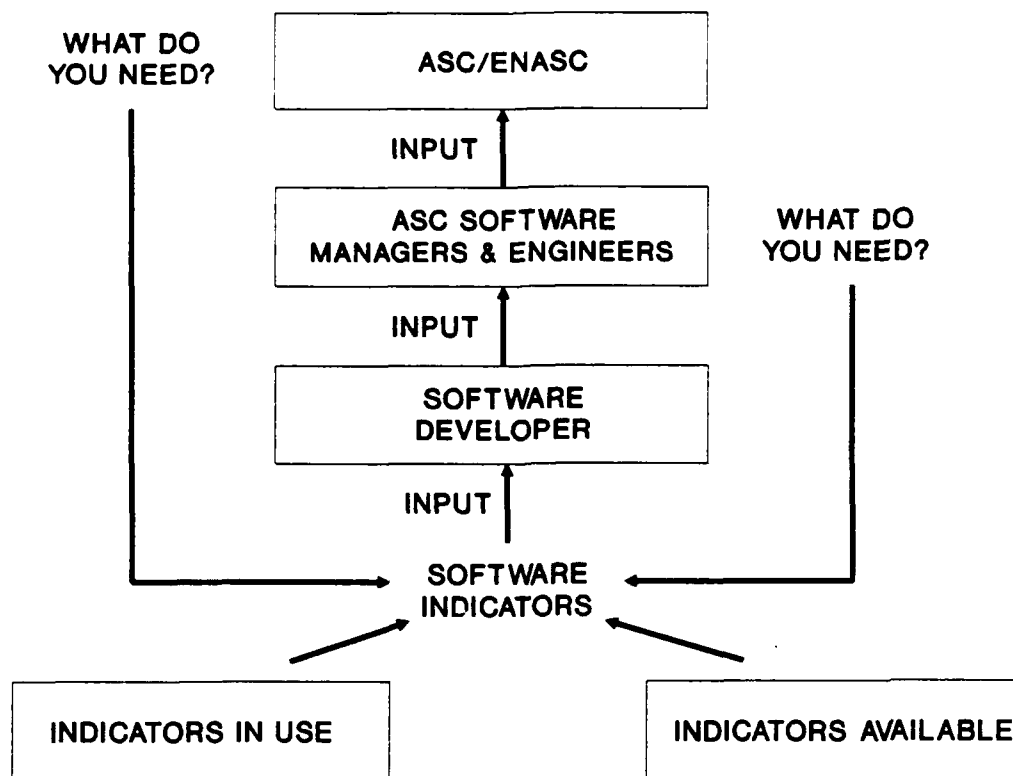


Figure 3.10 Research Flow Diagram

The research flow diagram was used as a basis for generating research questions to obtain information necessary to document the current software indicator environment. Documenting the process elements of the flow diagram provided knowledge of the current process which lead to the development of a standard set of software indicators for ASC.

It should be noted that though the scope of the research effort does not include activities that are accomplished outside of ASC, the software developer is included because of its importance in the software indicator process.

3.1.2 Developing Research Questions from the Research Flow Diagram. The objective of this task is to develop research questions to be used to document the research flow diagram. Because of a lack of knowledge of the current process, it was

decided the research questions should be open-response questions. Open-response questions are used when there is a lack of information about the process and the respondents who would be answering the questions (14:366-367). The research questions were generated by looking at the process flow diagram and formulating questions which would provide additional insight into each element of the process. To accomplish this, the research flow diagram was divided into two subprocesses: 1) the process the software managers and engineers were involved with, and 2) the process ASC/ENASC was involved with.

3.1.2.1 Developing Research Questions for the Software Managers and Engineers Subprocess. A flow diagram of the process the software managers and engineers were involved with was generated from the research flow diagram. This flow diagram is shown in Figure 3.11.

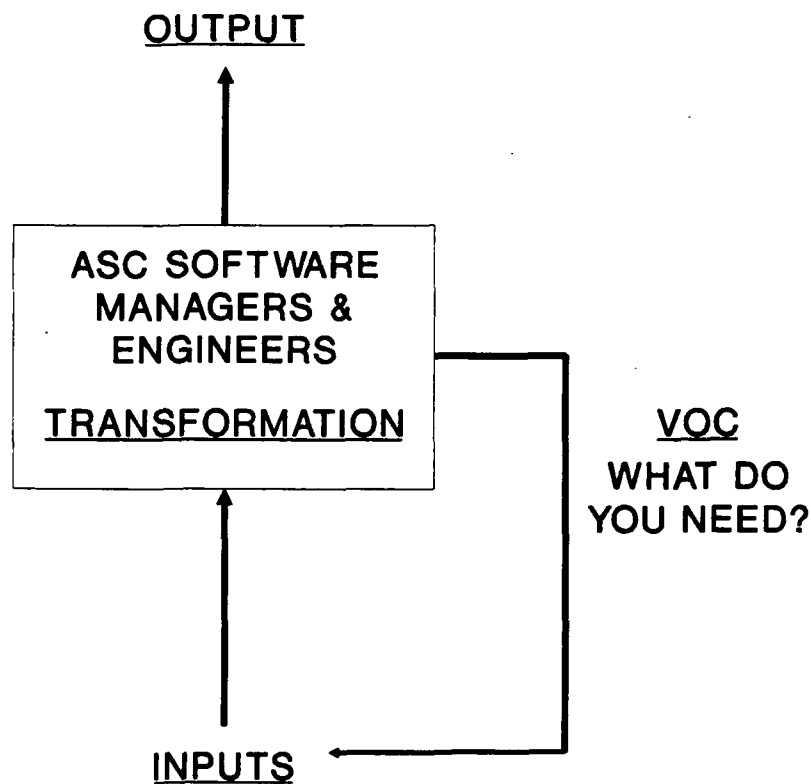


Figure 3.11 Software Managers and Engineers Subprocess

It should be noted that the voice of the process was not included in this flow diagram or the research flow diagram. The voice of the process would be represented by the software developers. This would be outside of the scope of the research effort which is restricted to efforts that are local to ASC.

There was little information known about the use of software indicators within ASC. However, it was known that some programs were using indicators and others were not. As a result, two sets of research questions were developed for the software managers and engineers: 1) a set for those programs that were using indicators, and 2) a set for those programs that were not.

Each set of research questions focused on the elements in the flow diagram of the process: inputs, transformation, outputs, and voice of the customer as represented by the software managers and engineers.

In addition to the process elements, process variables that could impact the use and selection of indicators were identified. These variables were divided into two categories: people and program. Research questions were also generated to obtain more information about the impacts of these variables on the process.

Table 3.1 shows the questions developed for the people and program variables, the inputs, transformation, outputs, and the voice of the customer. The questions are divided by whether they pertain to the interview of personnel with indicators, without indicators, or both.

TABLE 3.1  
SURVEY QUESTIONS

People Variables (both surveys):

Question: Is the individual a manager or engineer?

TABLE 3.1

SURVEY QUESTIONS (CONTINUED)

Rationale: Determine how the use of indicators is dependent on whether the individual is a manager or engineer.

Question: How many years have managers and engineers been involved with software development efforts?

Rationale: Determine if the use of indicators is dependent on the experience levels of the managers and engineers.

Question: What types of training have the managers and engineers had to prepare them for their jobs?

Rationale: Determine if the use of indicators is dependent on the types of training managers and engineers have had.

Question: How long has the manager or engineer been in their current job?

Rationale: Determine if the use of indicators is dependent on how long managers and engineers have been in their current job.

Program Variables (both surveys):

Question: When was the contract awarded for the software development effort?

Rationale: Determine if the use of indicators is dependent on when the effort was initiated.

Question: What phase of the system lifecycle is the software development in?

Rationale: Determine if the use of indicators is dependent on which phase of the lifecycle the software development effort is in.

Program Variables (with indicators surveys):

Question: Are indicators required by contract?

Rationale: For those efforts using indicators, determine if their use is dependent on whether or not the developer is required to deliver them.

Inputs (both surveys):

Question: What types of information do managers and engineers get to help them accomplish their job?



TABLE 3.1

SURVEY QUESTIONS (CONTINUED)

**Rationale:** Indicator data may not cover all information needs of the managers and engineers. Therefore, all sources of information must be defined so that inefficiencies in the indicator process can be evaluated. It also provides insight into what sources of information are used by managers and engineers who are not using indicators.

Inputs (with indicators surveys):

**Question:** By what process is software indicator data gathered?

**Rationale:** Used to determine if the managers and engineers receive the data. This information will be used to determine the dependency of the managers and engineers are on the software developer for data analysis.

Transformation (both surveys):

**Question:** What are the responsibilities of and functions performed by managers and engineers?

**Rationale:** The transformation of inputs to outputs should be in direct support of the job that each manager and engineer has.

Transformation (without indicators surveys):

**Question:** How do managers and engineers estimate/predict costs, schedules, and product quality?

**Rationale:** As stated above, some managers and engineers may not be using indicators. If they are not, then how they derive the information that indicators would provide is important to understanding if indicators would have any added value for their process.

Transformation (with indicators surveys):

**Question:** What happens to the indicator data?

**Rationale:** This question provides information on how managers and engineers transform the indicator data into outputs.

Outputs (both surveys):

**Question:** What types of information products do managers and engineers provide to higher-level management?

TABLE 3.1

SURVEY QUESTIONS (CONTINUED)

Rationale: This question provides information on what outputs managers and engineers produce.

Voice of the Customer (VOC) (both surveys):

Question: How does the information you get to help you in your job differ from what you actually need to do your job?

Rationale: Trying to focus on the current gap between what the customer needs and what the process is providing him/her.

Question: Is there anything else about the use of indicators that you feel is important that hasn't been covered?

Rationale: Obtain information from "customer" on any other thoughts they have on the use of indicators to get as much information as possible on their perceived needs.

Voice of the Customer (VOC) (with indicators surveys):

Question: What do you believe are the primary reasons why software indicators are being used?

Rationale: Obtains information from "customer" on motivation for using indicators.

Question: Have software indicators significantly influenced your ability to accomplish your job? Why/Why not?

Rationale: Obtains information from "customer" on whether or not the indicators are meeting his/her needs by influencing their ability to do their job.

Question: Have there been any changes to the software indicators you are using? If yes, why were changes made?

Rationale: Obtains information from "customer" on any dissatisfaction with indicators that may have resulted in changes to the software indicators they were using. Also, gives insight into the temporal nature of the software indicator process.

Question: Do you expect any future changes to the software indicators you are using? Why/Why not?

TABLE 3.1

SURVEY QUESTIONS (CONTINUED)

Rationale: Obtains information from "customer" on any changes they may be contemplating that would change their feelings towards software indicators and their uses. Also, gives insight into the temporal nature of the software indicator process.

Voice of the Customer (VOC) (without indicator surveys):

Question: Do you envision software indicators being used on your program in the future? Why/Why not?

Rationale: For those programs not currently using indicators, it provides information on possible changes in their process to move toward the use of indicators. Also, gives insight into the temporal nature of the process.

Question: What do you believe are the primary reasons why software indicators are not being used?

Rationale: Obtains information from "customer" on problems with the process that inhibit the use of indicators.

In addition to these questions, the research team included several other questions that supported the interview itself or provided information for the research effort. Table 3-2 shows these added questions.

TABLE 3.2

INFORMATIONAL SURVEY QUESTIONS

Information Questions (both surveys):

Question: What program are you currently working?

Rationale: Verify the program that the manager or engineer is working is same as on the Preliminary Data Sheet.

Question: What are your overall impressions of the interview?

Rationale: To discover ways to improve the interview for the future.

TABLE 3.2

INFORMATIONAL SURVEY QUESTIONS (CONTINUED)

Information Questions (with indicators surveys):

Question: Can you describe the process that was followed to come up with the indicators you are using?

Rationale: For those programs using indicators, to provide insight into what other processes are being used to select indicators in addition to this research effort.

Question: What are the primary reasons why indicators were not placed on contract?

Rationale: For programs using indicators that are not on contract, to gain insight to why this is the case. Information potentially could be factored into procedures to implement standard indicators.

Question: How did you motivate the contractor to accept the task of providing software indicator data?

Rationale: For those programs that have indicators on contract, it provides insight into any difficulties in getting the contractor to comply with the contractual requirement. Information will be used when developing procedures to implement a standard set of indicators.

Question: Have indicators significantly influenced your program?

Rationale: For those programs using indicators, to gain more insight to the effectiveness of using indicators.

3.1.2.2 Developing Research Questions for the ASC/ENASC Subprocess.

Research questions for the ASC/ENASC subprocess were constructed in the same way as those for the software managers and engineers subprocess. A flow diagram of the process that ASC/ENASC was involved with was generated from the research flow diagram. This diagram is shown in Figure 3.12.

From discussions with ASC/ENASC and their mission statement, it was evident that their "customers" are the managers and engineers at ASC. Because this information

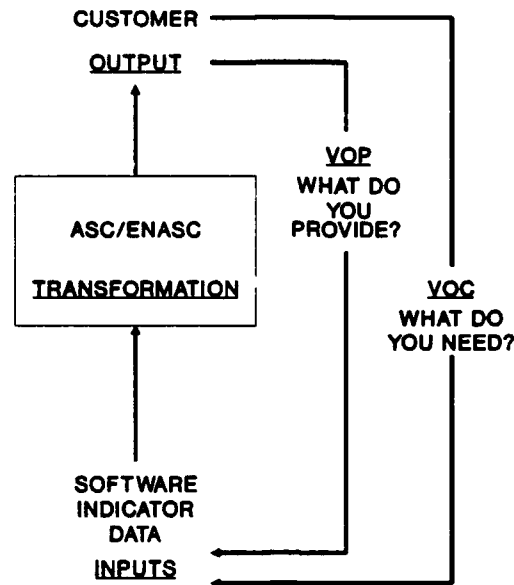


Figure 3.12 ASC/ENASC Subprocess

was known, and it was within the scope of the research effort, the voice of the customer is represented by the software managers and engineers. They are the real users of the products that ASC/ENASC will produce. Likewise, ASC/ENASC is shown as providing the voice of the process because they are responsible for producing the outputs that will be used by the software managers and engineers.

Between this flow diagram and the one that was generated for the software managers and engineers, there would appear to be a missing portion of the analysis. That "missing portion" is represented in Figure 3.13.

From reading ASC/ENASC's requirement and discussing it with them, it was understood that the inputs to this process would be software indicator data supplied by software managers and engineers. In reality, this data will be the same data that is used by the software managers and engineers. They will simply be passing the data to ASC/ENASC without doing any modifications to it. Because they will not be modifying the indicator data, the voice of the customer, represented by what the software managers

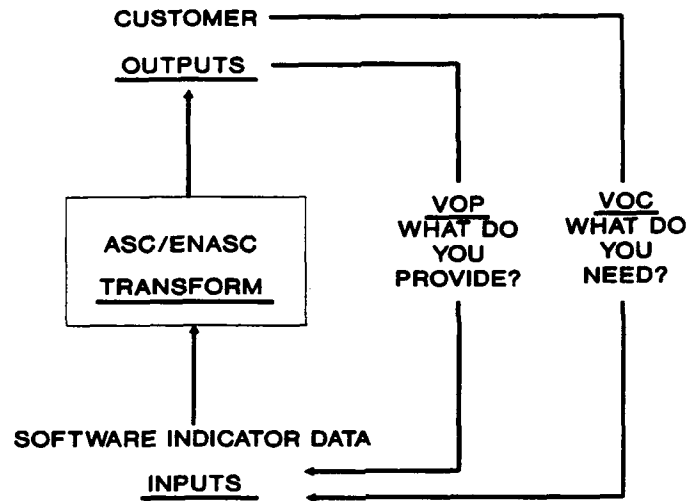


Figure 3.13 "Missing" VOC Relationship

and engineers need, really reflects the voice of the process, represented by what they will provide ASC/ENASC. Therefore, the comparison of the voice of the process, represented by what the software managers need, to the voice of the customer, represented by what ASC/ENASC needs, is already being accomplished by the development of a set of core areas which reflects all their needs. Therefore there was no missing portion of the analysis between the needs and what is provided, and the corresponding analysis will be addressed in Section 4.2.4.

It was understood that ASC/ENASC intended to incorporate the indicator data into a database. As a result, research questions were developed to obtain more information concerning the purpose of the database (representing the transformation element) and the outputs that would be produced. Research questions were also developed to obtain more information about the voice of the process and the voice of the customer. Table 3.3 provides the research questions that were developed along with accompanying rationale.

TABLE 3.3  
ASC/ENASC SURVEY QUESTIONS

Transformation:

Question: What are the primary objectives of the database?

Rationale: Provides insight and better understanding of what the database is to do.

Outputs:

Question: What do you expect to provide the customer?

Rationale: This question provides information on what the intended output of the database is to be. It provides a definition of the database outputs.

Question: How do you expect to provide outputs from the database?

Rationale: This question provides information on the actual form of database products that were identified in the previous question.

Voice of the Process:

Question: Who do you foresee as the "customer" of the database effort?

Rationale: Provides ASC/ENASC perspective (from the process) of who is the customer or user of the database.

Question: What do you believe the customer needs from the database?

Rationale: Looking for the voice of the process in terms of what ASC/ENASC feels is the voice of the customer.

Voice of the Customer:

Question: What did you need ASC/ENASC to provide you?

Rationale: Looking at defining the needs or voice of the customer.

Question: Were your needs satisfied?

Rationale: Looking at determining if there is a gap between the voice of the customer and the voice of the process.

TABLE 3.3

ASC/ENASC SURVEY QUESTIONS (CONTINUED)

Question: Is there anything else you would like to see ASC/ENASC provide?

Rationale: Further defining the voice of the customer to ascertain things that would reduce the "gap".

Question: Would you use the services provided by ASC/ENASC again?  
Why/Why Not?

Rationale: Further definition of the voice of the customer.

In addition to these questions, several questions were developed simply to provide background information and a better understanding of the database effort. These questions are shown in Table 3.4.

TABLE 3.4

ASC/ENASC BACKGROUND QUESTIONS

Question: Who originated the database effort?

Question: When was it originated?

Question: Why was it originated?

Question: Is there any formal documentation identifying the need for the database, validating it as a requirement, or providing tasking for it?

3.1.3 Developing a Survey Instrument. It was decided the best way to obtain the necessary information was a scheduled structured personal interview. The scheduled structured personal interview approach was selected because obtaining the needed information would require direct interaction with software managers, engineers, and ASC/ENASC. The intent of the interview was to define the current software indicator



environment. The use of open-response questions to gain as much information as possible about the process required obtaining information in great detail and over a broad range. The advantage of a personal interview is its ability to provide that type of information (14:320).

It was also decided that there were two distinct groups that would have to be interviewed because they were responsible for different parts of the research flow diagram. Those groups were: 1) software managers and engineers, and 2) ASC/ENASC. The next two sections describe how the interview questionnaires were constructed for each of these groups.

#### 3.1.3.1 Construction of Software Managers and Engineers Interviews.

Once the research questions were developed, they were incorporated into an interview questionnaire. To help the interview flow smoothly and avoid confusing the respondent, the research questions were reorganized into three logical groups. These groups were: 1) general information about the software development effort the respondent was working on, 2) the current software indicator environment, and 3) information about the respondent's job and what was needed to accomplish that job. Each of the research questions were included in one of these three groups. The groups then became separate sections of the interview questionnaire.

As stated previously, there were two sets of research questions developed. One set addressed programs that were using indicators, the other addressed programs that were not. Therefore, two separate interviews were constructed. The first and last sections, general information and job needs, were the same for both interviews. Only the current indicator environment section was different.

3.1.3.1.1 The General Information Section. The general information section verified which software development effort the respondent was responsible for,

when the contract for the effort was awarded, when it would terminate, and what life-cycle phase (i.e., Engineering and Manufacturing Development, Production, etc.) the software development effort was in. This information was gathered to ascertain any relationships between the age of the contract and whether or not indicators were being used. In addition, it could be used to determine whether there are different information needs for different phases of the development effort.

3.1.3.1.2 The Job and Information Needs Section. The job and information needs section began by determining the background of the respondent. Then questions were asked to determine what the respondent's job was, what information was available to aid in accomplishing that job, and what additional information was required. Additionally, questions were asked to determine what outputs the respondent generates as part of the job requirements.

3.1.3.1.2.1 Interview for Respondents Not Using Indicators. The first question in the current indicator environment section of the interview for programs not using software indicators asked for reasons why indicators were not being used. The respondent was next asked to describe the method that was currently being used to obtain the types of information that indicators would provide. That was followed by a question that asked if they envisioned any changes in the future concerning the use of software indicators.

3.1.3.1.2.2 Interview for Respondents Using Indicators. The first question in the current indicator environment section of the interview for programs using software indicators asked for reasons why indicators were being used. Then the respondents were asked questions concerning the contractual nature of the software indicators (i.e., were they required by contract, why or why not). Questions were then asked concerning the method that was followed to develop the set of

indicators being used, the process by which indicator data was gathered, what happened to the information provided by the indicators, and what influence the indicators had on the ability to accomplish the respondent's job and the software development effort in general. Finally, questions were also asked concerning changes to the software indicators being used, both in the past and future.

3.1.3.1.3 Construction of the Data Sheets. At this point, adequate questions had been developed to document the flow diagram and those questions had been incorporated into an instrument to gather the required information. However, this was not sufficient. In Chapter 2, the concept of operational definitions was discussed. Operational definitions take concepts and give them communicable meaning which can reduce the gap between the voice of the customer and the voice of the process (12:276). The current questionnaire with open-response questions would provide information to document the flow diagram, but it would not provide sufficient information to be able to operationally define the needs of the software managers and engineers. Therefore, in conjunction with the open-response questions, detailed information concerning the use of indicators and the needs of the software managers and engineers also had to be obtained.

Two things have to be taken into account when developing an operational definition: method of measurement, and criteria for judgement (25:16). In looking at this definition again, the research team came to the conclusion that the real operational definition of the needs of software managers and engineers are the software indicators themselves. As evidenced by the indicators in Appendix A., indicators certainly specify a method of measurement as details are specified on how the indicator is derived. They also provide a criteria for judgement as generally the indicator is comparing current data against an estimate or standard. Therefore, the data required to construct an opera-

tional definition of the needs of software managers and engineers is provided by the standard set of software indicators identified by this research effort. Therefore, a different set of data had to be obtained that would translate the needs of the software managers and engineers to a standard set of software indicators. The way this task was approached was similar to the method used by the Software Engineering Institute (SEI) when they developed their Capability Maturity Model for Software.

The Capability Maturity Model for Software consists of five maturity levels. Each level contains key process areas. Each key process area has key practices that are necessary for satisfying the goals of the key process area. Key indicators are those key practices, or components of key practices, that provide visibility into whether or not the goals of a key process area have been satisfied. (29:25-32)

The software development process, like a maturity level, contains several areas such as schedule, cost, testing, software performance, etc. Likewise, software indicators are used to provide insight into these same areas as shown by the indicators in Appendix A. Following the same process as the Capability Maturity Model for Software, from the software development process can flow key areas critical to software managers and engineers. From these key areas, software indicators can be identified to provide the software managers and engineers insight into whether or not their key areas are being addressed by the software developer. A way had to be determined to obtain detailed information which would identify the key areas of the software development process that are important to the software managers and engineers.

From the indicators in Appendix A, the following areas were identified:  
Schedule/Project Tracking, Reliability, Cost Projection/Estimation/Tracking, Software Performance, Software Characteristics, Quality, Manpower, Testing, Requirements Traceability/Stability, Maintainability, Documentation, and Software Process Consistency.

This was the set from which to derive key areas. The next step was to develop an instrument to determine which of these areas software managers and engineers would determine to be key areas. This was accomplished by generating two data sheets that respondents would be asked to fill out: a software areas sheet, and software indicator sheet.

3.1.3.1.3.1 The Software Areas Sheet. The software areas sheet contained the list of areas that was identified previously. The respondent was asked to rate each area in terms of how important that area was to accomplishing the respondent's job. A scale from 5 to 1 was used by the respondent to rate each area with 5 being "very important" and 1 being "very unimportant." This sheet would provide detailed information about which software areas are most important to software managers and engineers.

3.1.3.1.3.2 The Software Indicators Sheet. The software indicator sheet was developed to provide a cross-check to the areas sheet. It would also provide information concerning the level of familiarity that software managers and engineers had for software indicators. The indicator sheet contained the titles of 30 software indicators taken from the indicators listed in Appendix A. These indicators were picked to coincide with the areas that were contained in the areas sheet. This would allow for a cross-check between the two sheets. The respondent was first asked to determine if the indicator was or was not familiar to them. Then the respondent was asked to rate the indicators, using the same 5 to 1 scale that was used on the areas sheet, on how important they were to helping accomplish his/her job. This sheet would also provide detailed information about which indicators are most important to software managers and engineers.

3.1.3.1.3.3 The Software Publications Sheet. The software publications sheet was developed as a cross-check to the software indicator sheet. It contained the publications that were the sources of the indicators used in the previous sheet as well as other publications that address software indicators. Respondents were asked to simply put a "y" if they were familiar with the publication or an "n" if they were not. There would be a correlation between the indicators identified as familiar on the indicator sheet and the publications identified as familiar on this sheet. Additionally, respondents were asked to identify additional publications they were familiar with that were not listed on the sheet. This information may be used to identify additional software indicators not contained in Appendix A. This sheet would provide detailed information as to which publications were familiar to ASC software managers and engineers and correlate to the indicators that were identified as being familiar on the previous sheet. Figure 3.14 shows the relationship between the publications the indicators were taken from, the area that they correspond to, and the number of the indicator itself. The names of the indicators that correspond to the numbers in Figure 3.14 are as follows: 1) Computer Resource Utilization, 2) Fault Density, 3) Software Development Manpower, 4) Test coverage, 5) Testing Sufficiency, 6) Cost/Schedule Deviations, 7) Completeness, 8) Requirements Definition and Design Stability, 9) Schedule Progress, 10) Defect Density, 11) Memory Utilization, 12) Software Documentation and Source Listings, 13) Throughput, 14) Requirements Traceability, 15) Documentation 16) Deviation of End Product..., 17) Mean-Time-To-Failure, 18) Requirements Compliance, 19) Software Progress - Development and Test, 20) Defect Distribution, 21) Test Accuracy, 22) Software Size, and 23) Design Progress.

3.1.3.1.4 The Final Interview Questionnaire. The data sheets combined with the open-response questions provided a complete interview questionnaire which the

research team could now use to obtain the necessary information to document the process flow diagram and determine the needs of software managers and engineers.

In addition to the interview questions and data sheets, for each program that was using software indicators, the respondent was asked to provide a copy of the indicators. That would allow the research team to document which indicators were being used to expand the database of indicators that are currently available.

PUBLICATION	SW AREA	SW PROCESS CONSISTENCY	DOCUMENTATION	REQUIREMENTS	TESTING	MANPOWER	QUALITY	SOFTWARE CHARACTERISTICS	SW PERFORMANCE	COST	RELIABILITY	SCHEDULE
AFSCP 800-43	19			8		3		22	1	6		
IEEE 982.1			12	14 18	21						17	
ESD TR 88-01	9 23											
AFSCP 800-14			7 15		4 5		2 10					
SEI 12-1.1												
AFSCP 800-49							20		11 13			

Figure 3.14 Indicator/Area/Publication

3.1.3.1.5 Validating the Interview Instrument. The external validity of the interview instruments was initially checked by having the thesis committee and ASC/ENASC review them and provide comments. Additionally, the first five interviews were used as a check for the external validity of the questions and to ascertain the need for changes for future interviews.

The internal validity of the interview instruments was checked primarily by the cross-checking of the data sheets and the comparison of the responses to the ability to

characterize the indicator environment. Again, the first five interviews were used to determine internal validity.

In addition, the areas and indicators sheets were given to the respondent at different parts of the interview. The software indicator and publication sheets were given to the respondent to fill out during the current indicator environment section of the interview. The software areas sheet was withheld until the job and information needs section of the interview. By doing this, the respondents would not bias their responses to the areas sheet based on what they responded with on the indicator sheet.

3.1.3.2 Construction of ASC/ENASC's Interview. Once the research questions were developed, they were incorporated into an interview questionnaire. There were two questionnaires developed: one for ASC/ENASC, and the second for the software managers and engineers who use ASC/ENASC's services.

The ASC/ENASC questionnaire contained the questions concerning transformation, outputs, and voice of the process. The research team also included the areas sheet that was developed for the software managers and engineers interviews in the ASC/ENASC questionnaire. This would provide detailed information about which areas ASC/ENASC felt were key to the database effort.

The "voice of the customer" research questions were incorporated into a questionnaire for the software managers and engineers who use ASC/ENASC's services.

3.1.4 Conducting the Interviews. This section covers the actual gathering of the data for the research effort. There were three types of interviews conducted: 1) the interview of software managers and engineers, 2) the interview of ASC/ENASC, and 3) the telephone interview ASC/ENASC customers.



#### 3.1.4.1 Conducting the Software Managers and Engineers

Interviews. These interviews were conducted from March through May 1992 at Wright-Patterson Air Force Base, Ohio.

##### 3.1.4.1.1 Obtaining the Sample of Individuals to Interview.

To conduct the scheduled structured personal interviews, a sample of software managers and engineers within ASC had to be selected. The population that the sample came from was all the software managers and engineers within ASC. Initially, a sampling frame of software managers and engineers was developed by contacting each ASC organization's Computer Resource Focal Point. A list of the Computer Resource Focal Points was obtained from ASC/ENASC. A meeting was scheduled with each focal point to discuss the research effort, obtain any feedback, and request their support for it. In addition, each focal point was asked to provide a list of software managers and engineers for each software development effort in their organization. It is from this sampling frame that the sample of software managers and engineers was taken.

##### 3.1.4.1.2 Scheduling the Personal Interviews.

For a personal interview to be successful, three conditions must be addressed: the information needed from the respondent must be available, the respondent must understand their role, and the respondent must be motivated to respond (14:321). One way that these conditions were addressed was by working through the computer resource focal points within each organization to identify individuals to be interviewed. It was hoped that the focal points would address these factors during that identification process. The other way these conditions were addressed was by using a Preliminary Data Sheet to schedule the interviews with individuals (Appendix B). The Preliminary Data Sheet was used to explain to individuals the purpose of the research effort, what the general structure of the interview was, and obtain preliminary data from them. The first piece of data that

was obtained from the individual is whether or not they were willing to support the research effort. Other data consisted of whether or not software indicators were currently being used, whether or not the individual was a manager or engineer, and what program they were involved with. Another way that individuals were motivated to participate was by keeping the responses to the interview anonymous. The interest of this research effort was to obtain a global view of ASC and not single out individuals or individual software development efforts. Of the thirty-four individuals contacted from the sampling frame, every individual agreed to participate in the interviews. Of these, twenty-four were engineers, nine managers, and one individual whose job appeared to be in both areas. There were thirteen military and twenty-one civilians interviewed. There were twenty separate programs these individuals were working on, and each two-letter office at ASC had at least one program interviewed for the research. The respondents were also asked about job experience and software experience. The average amount of job experience was 3.4 years, with a standard deviation of 2.4 years and a sample maximum/minimum of 12/0.5 years. The average amount of software experience was 8.4 years, with a standard deviation of 6.8 years and a sample maximum/minimum of 30/0.25 years.

3.1.4.1.3 Changes to the Interview Questionnaire. As stated previously, the first five interviews were going to be used as validity checks for the interview. From these interviews, the research team concluded that the interview questionnaire was a valid data gathering instrument. However, some minor changes to the questionnaire were made based on feedback obtained during these interviews.

The first change was to add a question to the general information section to determine when the contract was to end. The rationale for this change was to determine

if contracts were nearing completion where it would not make sense to put indicators on contract. In addition, it provided the contract length as another program variable.

The second change was to add a question to the software indicator sheet to ask the respondent to rate how important they felt software indicators to be. The rationale for this change was to ascertain if the individual personally felt that software indicators were important enough to implement.

Another change that was made was to add a question to the general information section to ask the respondent the type of contract they were working on (i.e., firm fixed price or cost plus). The rationale for this change was because the use of indicators may be influenced by the type of contract, i.e. cost indicators may not be as important on firm fixed price contracts as they are on cost plus contracts.

A fourth change that was made was to the questionnaire for those programs with indicators. For the question that asked if any changes were expected to the indicators in the future, the respondent was asked to provide reasoning for a negative response. The rationale for this was that it may provide important information concerning the implementation of a standard set of indicators.

After 13 interviews were conducted, a meeting was held with the thesis committee to discuss the progress of the data gathering effort and review some preliminary data analysis. Based on the discussions from this meeting, changes were made to the indicator and area sheets.

The results of the preliminary data analysis showed there were seven indicators in the indicator sheet that no one had rated as very important. It was determined that these indicators could be deleted from the sheet to allow more time to accomplish the interview.

For the areas sheet, it was determined that instead of rating each area the respondent would be asked to rank order all the areas. The research team reviewed a report on a survey that was done on software indicators. In this survey, respondents were asked to both rate and rank indicators. The report showed that there was a difference between the rating and ranking of the indicators. It was believed that by having the respondent rank the areas it may have produced a better assessment of which areas are really key areas.

In addition to these changes, two additional questions were added to the general information section: is cost performance required for the contract, and what is the estimated size of the software? The rationale for these were that cost performance reporting may negate the need for a cost indicator, and smaller programs may not need software indicators.

Another change was made after the fourteenth interview. The areas sheet was changed back to having the respondent rate instead of rank the areas. It was discovered that it was too difficult for the respondent to try to rank the 12 areas that were on the sheet. However, in place of the ranking, respondents were asked to select a maximum of five areas that they felt were the most important areas.

One last change was made to the questionnaire after the sixteenth interview. The indicator sheet was changed to have the respondent briefly discuss their rationale for rating indicators as very important (5), unimportant (2), or very unimportant (1). The rationale for this was to gain greater insight into why some indicators were rated high and others rated low.

3.1.4.2 Conducting the ASC/ENASC and Customer Interviews. ASC/ENASC was contacted to establish a time to conduct the interviews, and they were completed. ASC/ENASC was then asked to provide a list of individuals

that had requested their support. These individuals were then contacted by telephone and asked the "voice of the customer" questions.

3.1.5 Result of Defining the Current Environment. At this point in the methodology a research flow diagram has been developed, a survey instrument developed, and the survey carried out. This is shown in Figure 3.15.

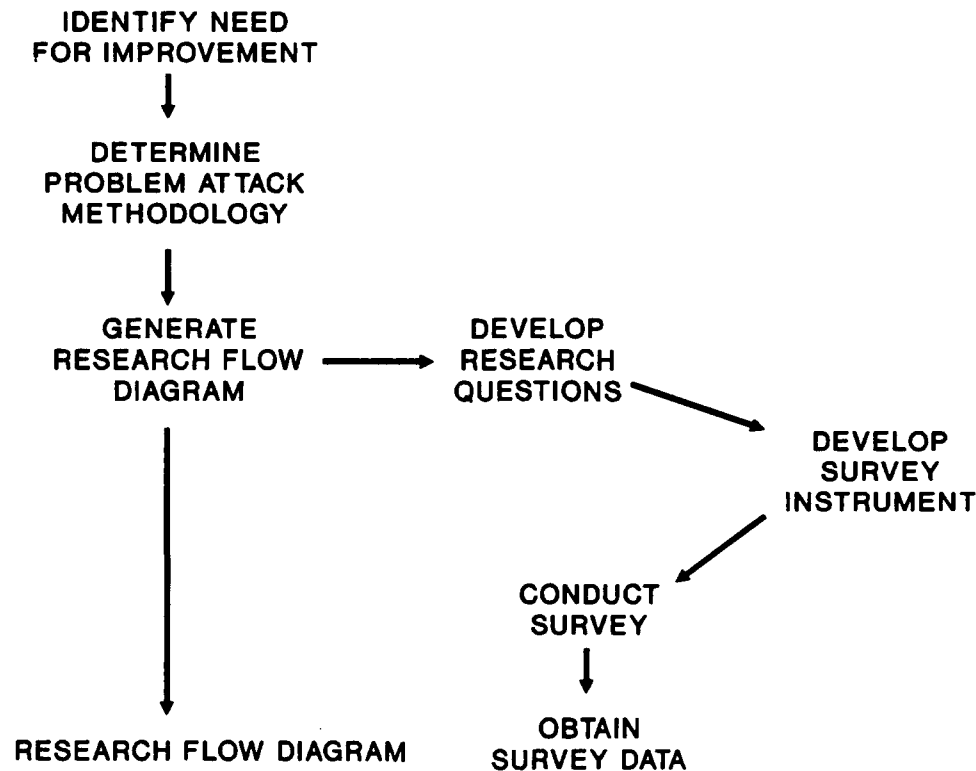


Figure 3.15 Defining the Current Environment

### 3.2 Documentation of the Current Environment.

Once the interviews were conducted, there were four sources of information available to support data analysis. These four sources of information were: 1) the research flow diagram; 2) responses to the process questions (input, transformation, output, etc.); 3) the responses to the questions dealing with variables (people and program); and 4) the responses to the area, indicator, and publications sheets. The

responses to the questions and sheets were entered into a database to enable sorting and manipulation to support the data analysis. The contents of the database is contained in Appendix D. The objective of this section is to document the current environment from this information. The methodology to accomplish this is shown in Figure 3.16.

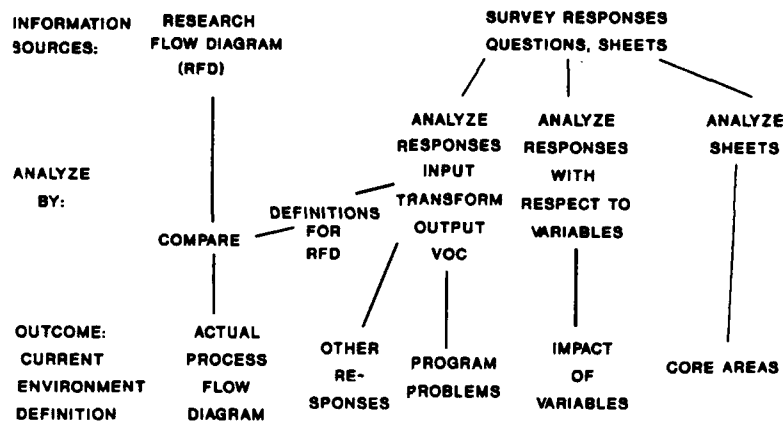


Figure 3.16 Data Analysis Process

3.2.1 Question Response Analysis. The first operation to be done on the data is to analyze the survey responses with respect to the basic process areas: input, output, transformation, and voice of the customer. The input to this operation is the survey responses. The objective of the operation is to define the software indicator environment process at ASC, in the categories of the basic process areas, identify problems that specific programs may have encountered, and identify examples of concepts being used on some programs.

3.2.1.1 Question Response Analysis Methodology. The methodology for this analysis was for the research team to study the total responses for each survey question. The analysis of the questions was done in five steps: 1) response verification;

2) listing/combination of responses; 3) distribution determination; 4) trend analysis; and 5) separation of statements identifying problems from the remainder.

Each response to the question was examined to ensure that it was indeed replying to the question asked (a danger with the open-response type of question).

A list of all pertinent responses was developed, and like responses were combined. An example of this combination would be that "program reviews" and "PMRs" and "quarterly reviews" would be combined under the heading "reviews". From the list of responses, the relative frequency of the responses was calculated.

A composite picture was the goal of the analysis. The composite response was determined by one of three methods: 1) a Pareto analysis (see Section 3.2.2.1 for a description of a Pareto analysis), 2) grouping, or 3) diverse response. The composite picture consisted of the one or two significant responses from all the responses. A Pareto analysis was performed on the responses first. If this was unsuccessful, the next step was to try to group responses into broader categories. Examples of this might be "telephone calls", "plant visits", and "E-mail" being combined into a group called "informal contact with contractor". These broad groups provided some insight into the process, but not as detailed as the Pareto. If the sample size was too small to use a Pareto or group analysis on, and the responses are all different (e.g., four unique responses out of a four total responses), then the responses were considered diverse, no combining was required. Significant responses were then listed for an analysis of the research flow diagram in a later analysis.

Responses at this time were also examined for problems in each program with the software indicator environment. The responses of all the survey questions were examined, as the open-response type of questions on the surveys can lead to information at any point. Examples of program problems might be "indicators are not used on this

software development due to cost" or "my boss ignores the indicators I show her".

Criteria of analysis for this area is: Indication in a survey response that there exists a problem with the use of indicators in the interviewee's software indicator environment.

Remaining responses were categorized as "other responses". These responses are used in the improved environment analysis as a potential source of solutions to process and program problems identified in the examination of the process.

3.2.1.2 Question Response Analysis Outcomes. The product of this analysis was the process definition responses that had a potential to impact the research flow diagram, program problems and good examples that were culled from the responses. The process definition responses will be used in Section 3.2.4 for the analysis of the research flow diagram.

3.2.2 Analysis of Software Managers, Engineers, and ASC/ENASC Indicator and Area Preferences. This analysis is divided into sections for each of the indicator and sheets. Managers, engineers, and ENASC are used as the primary categorization of indicators and areas as the needs of the three groups was seen by the research team as potentially being significantly different. Indicator, area, and publication sheets are analyzed separately. The objective of this analysis was to identify the core areas for managers, engineers, and ASC/ENASC.

3.2.2.1 Indicator Sheet Analysis Methodology. Indicator sheets are analyzed to determine what indicators are viewed as significant by software managers and engineers. Multiple criterion are used to determine the level of significance for each indicator. Specifically, three types of reduction was done to the indicator data: 1) majority analysis, 2) Pareto analysis, and 3) weighted average.

The first type of data reduction is to look at which indicators received a majority of very important, or "5" ratings based on the number of respondents who rated that



particular indicator. Only those indicators that received a majority rating, 51 percent or greater, are identified for use in selecting the core areas. The rationale for using only "5" ratings is that the "very important" ratings reflected those indicators that were absolutely necessary in the minds of the interviewees and therefore the areas they represent should make up the core areas from which indicators would be derived.

The second type of data reduction is a Pareto analysis for software managers' and engineers' indicator sheets. The analysis concentrated on the number of very important ratings an indicator received. The same rationale stated above for only using "5" ratings applies in this case also. A Pareto analysis results in the formation of three categories or classes of items. These categories are represented in Table 3.5.

TABLE 3.5  
PARETO ANALYSIS CATEGORIES

<u>Category</u>	<u>% of Total Items</u>	<u>% of Total Output</u>
A	10	70-80
B	10-20	10-15
C	70-80	10-20

(21:152)

It is noted that these percentages may vary somewhat from organization to organization (21:152). Category A indicators are considered significant in this study.

The third type of data reduction is a weighted average rating. Indicators are also ranked according to the average rating for each indicator, to determine the importance of indicators with respect to ratings of less than "5". A rating more than 4.5 for an indicator were considered significant.

Indicators that are shown by these analyses to be of significant impact to software managers and engineers are then matched to the software indicator areas they correspond to, and this analysis provides one input into the selection of core areas indicators will be drawn from for measurement of the software indicator environment.

3.2.2.2 Area Sheet Analysis Methodology. Area sheets are analyzed to determine what software indicator areas are viewed as significant by software managers and engineers, and ENASC. Multiple criteria are used to determine the level of significance for each area. These criteria were the same as those used in the indicator sheet analysis.

3.2.2.3 Publication Sheets Analysis Methodology. The publications sheet contained a list of eight Air Force, DOD, other service, MITRE, and commercial publications, regulations, and standards. The publications were the sources of the indicators on the indicator sheet. Respondents were asked to indicate if they were familiar with the publication. The responses to this sheet were then used to determine the level of familiarity of the software managers and engineers with publications on indicators. The results also were used to analyze whether the indicators rated on the indicator sheet were being selected based on familiarity with the indicator as published, or if the interviewees were responding to the name of the indicator alone, and using their rating as a show of approval for the area the indicator represented.

From the analysis of the indicator and area sheets, a set of key areas considered significant for software managers, engineers, and ENASC is developed. A core set of areas would be those areas that are analyzed to be significant in all three groups by the analysis above, the intersection of their indicated needs. If no areas were common to all three groups, the core set was to be derived by two alternate criteria: areas that were key

areas to two of the three groups, and areas that were considered critical to the function of a particular group.

3.2.2.4 Outcome of the Indicator and Area Preference Analysis. The result from this analysis is a set of core areas that are essential to the managers, engineers, and ASC/ENASC. These core areas will be used as the basis for the selection of a standard set of indicators in Section 3.3.1.

3.2.3 Analysis for the Impact of Variables. There were three objectives of the analysis of the impact of variables on the responses: 1) to determine if there are categories of people or program variables that affect the need or use of indicators, 2) if the variables had an impact on the areas considered very important by the respondents and 3) to establish a profile of the people and programs surveyed for the research. There was two phases to the analysis of the variables, the first phase being a comparison of the programs having indicators to those that do not have indicators, and a second a factoring of the variables into the analysis of the data sheets to see if the variables had an impact on the responses to the indicators and areas. Examples of these might be "SPOs with Z had indicators, and those without Z did not" or "SPOs with W believed software performance important, and those without W did not". Input to this analysis are the survey responses and the area sheets.

3.2.3.1 Impact of Variables Methodology. The variables were chosen using the software indicator environment model postulated at the start of the research and the hypothesis that these variables would have the most significant impact upon that process. The areas identified were then incorporated into the survey instrument to gather the data necessary to determine if these variables did indeed have an impact on the responses of the interviewees. The variables are summarized in Table 3.6.

TABLE 3.6  
PROGRAM VARIABLES

Type of contract: Firm fixed price or cost-plus

Size of software development: 100,000 lines of code and smaller or greater than 100,000 lines of code.

Contract length: Contracts of five years duration and shorter or contracts longer than five years.

Type of system being acquired: Embedded systems (those on a physically constraining platform) or mainframe systems (those at a fixed-location).

Program phase: Three categories, engineering/manufacturing development, or production, or a "combination" where the total production is only one or two units.

Cost performance reporting: Acquisitions that have cost reporting on contract or do not have cost reporting on contract.

Function: Work classification of interviewee as a manager or an engineer.

Criteria for the phases of analysis is in two steps. Step one is the comparison of the programs without indicators to the programs with indicators to see if there was a variable that could be identified as having a causal relationship to the presence or lack of indicators. Step two is done by dividing indicator and area sheets into two parts using one of the variables to differentiate them. Scores are then compared using the same methodology that the indicator sheets used. This analysis was done to determine if the variable could be construed to have an impact on the importance of an indicator or area as shown by the score given it by interviewees. The process was then repeated for each variable.

3.2.3.2 Outcome of Analysis of Impact of Variables. A list of variables that had an impact on the use of indicators, and a list of variables and the areas that they had an impact upon. Variables that are shown to have an impact on the indicators

are used to provide recommendations as to the tailoring of the standard set of indicators for a particular situation where the variable is shown to cause a difference. These tailoring recommendations are reported in the research findings and incorporated into the software indicator handbook (Appendix G).

3.2.4 Analysis of the Research Flow Diagram. The object of this analysis is to evaluate the research flow diagram in light of the data that was gathered about the process, and determine if the diagram required revision. The input to this analysis is the research flow diagram created to guide the construction of the survey instrument, and the distilled responses from Section 3.2.1.

3.2.4.1 Flow Diagram Analysis Methodology. The method of analyzing the research flow diagram was to combine the responses from the survey (Appendix D) with the flow diagram, placing the responses in the appropriate section (input, output, or transform) of the proper process (ASC software managers and engineers, ASC/ENASC). An additional telephone survey was conducted (Appendix D.4) to gather information on recent users of ENASC's services. Responses that identified actions or interdependencies that were different from those indicated in the research flow diagram identified areas that required modification.

3.2.4.2 Flow Diagram Analysis Outcome. The outcome of this analysis is a process flow diagram that reflects the actual software indicator environment at ASC.

3.2.5 Outcome of the Documentation of the Current Environment. The outcome of this documentation was a process flow diagram, a list of indicator problems of programs, a list of other responses about programs, a list of the variables that impact the use of indicators or the indicators areas considered significant, and a list of the core areas as shown in Figure 3.17. The process to obtain these results consists of four steps:

- 1) analysis of the question responses; a determination of the important areas for

software managers, engineers, and ASC/ENASC; an analysis of the impact of program and people variables on the use of indicators; and an analysis of the research flow diagram in light of the data gathered. This information satisfies research objective number one.

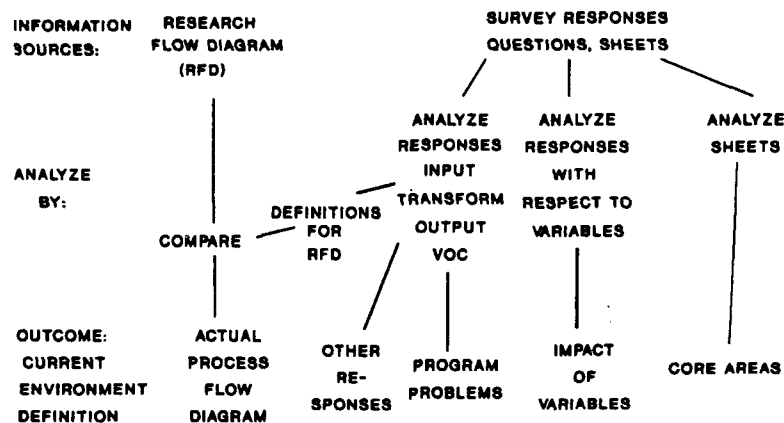


Figure 3.17 Data Analysis Methodology

### 3.3 Conceptualization of the Improved Software Indicator Environment.

The first step in the data analysis was to study the responses and to create a clear picture of the current environment. The next step is to take the products created in the documentation of the current process and create a vision of the improved indicator environment. This improved environment has two primary sections: the improved process to use indicators, and the standard set of indicators for that process. The method used to outline the improved process is to discover problems with the current process. This methodology is shown in Figure 3.18.

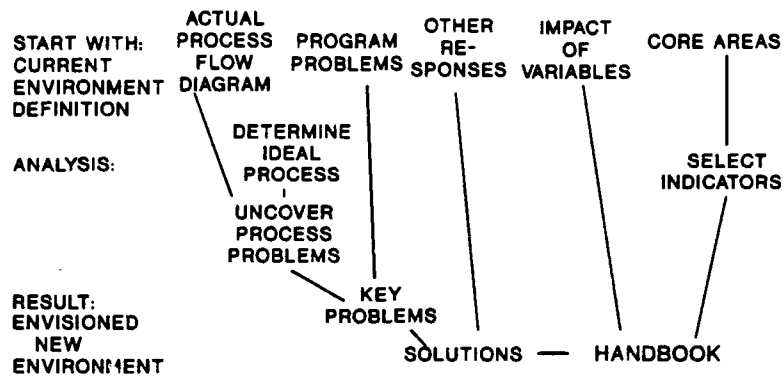


Figure 3.18 Improved Environment Methodology

3.3.1 Selection of the Standard Set of Indicators. The objective of this analysis is to determine what indicators best represent the core areas of concern for software indicators. The inputs to this analysis are the areas identified, the indicators from the various programs provided by the software managers and engineers, and the list of indicators compiled from the literature review of the research team.

3.3.1.1 Methodology for the Selection of the Standard Set. The method for selecting the standard set of indicators has four parts: 1) gathering a list of all indicators that apply to the core areas identified, 2) select an indicator for each area, 3) consider any optimization of the indicator that may be needed, and 4) develop procedures based on the source of the indicator for its implementation. Since the indicators from the literature had been compiled into a database prior to this effort and the indicators from the program were already collected and sorted on area, the first action to be done in this analysis is the comparison of the indicators.

After the identification of the core areas, a standard set of indicators is determined from the indicator database and the indicators in use by software managers and engineers that they have provided to the research team. Criteria are based on the

research goal of selecting the best, least costly indicators available for the core areas.

The criteria for selection is in Table 3.7.

TABLE 3.7

INDICATOR SELECTION CRITERIA

Utility. Indicator is examined for usefulness when compared to other candidate indicators

Clarity. Indicator is examined for ease of understanding when compared to other candidate indicators.

Complexity. Indicator is examined for its complexity in calculation (impacting difficulty of generation of indicator and as an indirect indication of cost) to other candidate indicators.

From this selection process, a standard set of indicators is developed.

During the selection process, certain aspects of some candidate indicators may be selected to optimize as many of the selection criteria as possible. The resultant indicator would be an amalgam of the characteristics of the indicators used to create it.

3.3.1.2 Development of Procedures for the Use of the Selected Indicators.

The procedures for the use of the selected indicators were taken directly from the source of the indicator. The procedures for use were modified appropriately for those indicators that were optimized.

3.3.2 Analysis for the Discovery of Process Problems. Analysis of the process to discover the process problems builds on the process flow diagram of the software indicator environment. The objective is to uncover those problems with the current software indicator environment that are a result of the process as a whole. The method for accomplishing this is in four steps: 1) develop an "ideal" software indicator environ-



ment, 2) compare this environment to the analyzed survey responses derived previously, 3) identify deviations in these responses from the ideal responses, and 4) examine these deviations to determine if they are caused by a flawed process.

3.3.2.1 Creation of an Ideal Software Indicator Environment. The objective of this step is the development of an "ideal" software indicator environment. This ideal environment is required to perform the analysis of the responses to the surveys. The input to this process is literature containing the definition of the responsibilities of the software managers, engineers, and ASC/ENASC. The first part of the methodology will consider the development of the ideal process for the software managers and engineers, and separate second part will discuss the development for ASC/ENASC.

The next step is to provide a methodology for the development of an ideal software indicator environment process definition. During the development of a standard set of software indicators, survey questions were asked to characterize the software indicator environment at ASC. Performing an analysis on the responses received requires a standard to compare them against. With such a standard, responses can then be examined to determine if the process the interviewee was involved in was in agreement with the standard or not. The ideal software indicator environment requires the development of expected responses for the survey instrument questions with respect to indicators. These questions are broken into the four parts of the process: inputs, transformations, outputs, and the voice of the customer. The first step in comparing the responses received to the standard is a definition of the standard for the four parts, then from these standards the "responses" of can be determined, and with these the comparison with the actual responses can begin. The most important part of the process is the

transformation, so we will look at the development of the standard for that first, then the inputs, outputs, and the voice of the customer.

For the development of a standard program management and program engineering transformation definition the research team was required to determine the types of functions that are the accepted norm for software managers and engineers. As stated previously in this chapter, a hypothesis of the research team was that software managers and engineers would have differing functions performed, and therefore potentially require different indicators.

The source used for the definitions of the functions of managers and engineers was the textbook for Air Force Institute of Technology's Systems 100 course, "Introduction to Acquisition Management". The course is required by the Air Force as part of the certification of acquisition personnel in the Acquisition Professional Development Program. Use of this source provided a definition that was widely recognized and (at least tacitly) approved by the software managers and engineers at ASC, as many of them had taken the course and been exposed to the definitions.

The source of the definition for the management functions is the extracted definition of the project manager's job in the projects division of a System Program Office (SPO):

Their function is to aid the program manager in coordinating the various activities to try and keep the program within cost, schedule, performance, and support constraints. The projects division normally interacts with all of the other divisions to gather information for program manager decisions and to coordinate the activities between the other functional divisions. (4:57)

The source of the definition for the management functions is the extracted definition of the engineering division of a SPO:

Engineering is responsible for overseeing the contractor's efforts in terms of the contractor's technical efforts, verifying that the contractor is performing the technical work correctly, and insuring that the various contractor engineering specialties are working in a coordinated manner to meet the user's requirements. (4:57)

The derivation of the ideal input to the software indicator environment builds on the transformation definitions established above. It takes the form of the simplest inputs of indicator data to the process, from the accepted baseline documents the software managers and engineers are required to use, as shown in Table 3.8.

TABLE 3.8

STANDARD INPUTS TO SOFTWARE MANAGERS AND ENGINEERS

1. AFSCP 800-43 specifies the data item descriptions to use in obtaining information that is gathered by the contractor, indicators derived from them, and the contractor generates a report of these indicators to the SPO.
2. Information is delivered to the SPO in formal documents required on all developments by MIL-STD 2167A: at program reviews, software status reports, preliminary design reviews, critical design reviews, and contractor data requirements list items.
3. From the transformations defined above, informal contact between the SPO software managers and engineers and the contractor is unavoidable. Information can be passed to the SPO in during these contacts.

The derivation of the ideal outputs of the software indicator environment builds on the transformation definitions established above. The outputs for the managers would be in the form of the assistance for the program managers decisions, and reports on what the managers have been involved in coordination of the "various activities" identified above. For the engineers, a similar situation exists in the "overseeing of contractor's efforts" and validation of those efforts.

The definition of the voice of the customer is not significant to the "ideal" software indicator environment. The Deming process model assumes that the significance of the voice of the customer is in the "gap" between this voice and the voice of the

process itself. In the case of the ideal process, the voice of the customer is entirely congruent with the voice of the process, since there is no need to change this process. This the voice of the happy customer that has no complaints about the process that is serving her.

3.3.2.2 Deriving the Standard Responses to the Survey Questions. Now that an "ideal" process has been developed, the survey questions can be examined in light of the elements of this process to derive the standard response set to which the response data collected can be compared. These responses will all be centered around the use of indicators for data gathering. The analysis is in the sequence of the information flow through the process; inputs, transformations, outputs, and voice of the customer. Table 3.9 shows this analysis, the question will be quoted from the survey and the ideal response given for those questions that directly impact these parts of the process.

TABLE 3.9

IDEAL SURVEY RESPONSES (MANAGERS AND ENGINEERS)

Inputs:

Question: By what process is software indicator data gathered?

Ideal Response: Indicator primitives are gathered by the contractor, indicators derived from them as specified in the contract, and I get a report of these indicators. I also receive information in informal contacts with the contractor.

Question: What types of information do managers and engineers get to help them accomplish their job?

Ideal Response: Besides indicator data, there is also information provided to me in formal documents required by MIL-STD 2167A; at program reviews, software status reports, preliminary design reviews, critical design reviews, and contractor data requirements list items.

TABLE 3.9

IDEAL SURVEY RESPONSES (MANAGERS AND ENGINEERS) (CONTINUED)

Transformation:

Question: How do managers and engineers estimate/predict costs, schedules, and product quality? (survey of SPOs not currently using indicators)

Ideal Response: I would use the information input to me.

Question: What happens to the indicator data?

Ideal Response (managers):

Indicator data is used as shown in DODD 5000.2 which:  
... requires a program manager to identify and use a set of software management indicators to capture key data elements, reflect the status of software development, augment conventional reports, and highlight areas that require attention. (800-43:3)

Question: What are the responsibilities of and functions performed by managers and engineers?

Ideal Response: Ideal response would be the software managers and engineers functions described in the definition of the transformation for managers and engineers.

Outputs:

Question: What types of information products do managers and engineers provide to higher-level management?

Ideal Response: I provide analyses of the indicator data received from the contractor and identification of the potential problems identified in the data. I provide status reports for program review efforts as required by ASC regulation.

Voice of the Customer (VOC):

Question: What do you believe are the primary reasons why software indicators are being used?

Ideal Response: Indicators are being used to provide the information necessary to perform the functions outlined in my job description, including to show the development status with respect to cost, schedule, performance, and support activities, identifying potential problems in these areas to support my corrective actions and provide the information needed to advise the program manager in his decision making process.

TABLE 3.9

IDEAL SURVEY RESPONSES (MANAGERS AND ENGINEERS) (CONTINUED)

- Question: What do you believe are the primary reasons why software indicators are not being used? (Survey of SPOs not currently using indicators.)
- Ideal Response: (There is no ideal response, and any responses provided would indicate a deviation from the ideal.)
- Question: Have software indicators significantly influenced your ability to accomplish your job? Why/Why not?
- Ideal Response: Indicators have influenced my job by providing the data necessary to support the functions outlined in my job description, and as outlined in DODD 5000.2.
- Question: Have there been any changes to the software indicators you are using? If yes, why were changes made?
- Ideal Response: There have been no changes to the indicators I am using. All indicators in use deliver the required data.
- Question: Do you expect any future changes to the software indicators you are using? Why/Why not?
- Ideal Response: No. I get all the information required from the indicators in use.
- Question: Do you envision software indicators being used on your program in the future? Why/Why not? (survey of SPOs not currently using indicators)
- Ideal Response: The ideal process would use indicators as defined by this research.
- Question: How does the information you get to help you in your job differ from what you actually need to do your job?
- Ideal Response: There is no difference. I get all the information I need from indicators.
- Question: Is there anything else about the use of indicators that you feel is important that hasn't been covered?
- Ideal Response: No.

3.3.2.3 Data Analysis Methodology for ASC/ENASC. The data analysis of the ASC/ENASC responses was carried out in the same manner as the software

managers and engineers. The standard response used was the responses from the four recent customers of ASC/ENASC. The four were identified by ASC/ENASC.

Development of a standard process model for ASC/ENASC begins with the examination of the research process diagram, Figure 3.10. This figure shows the inputs to the ASC/ENASC process. From this figure we see that the software managers and engineers provide the inputs to ASC/ENASC. The basic research question for the process at ASC/ENASC is shown in Figure 3.12, subprocess representation. In the memo that initiated the research, it is stated that the process that ASC/ENASC is trying to accomplish is to establish a "standard set of indicator definitions and tools . . . to provide consistent information at ASD." (5:1) A search of the ASC regulations and organizational documentation revealed no standard characterization for the process that ASC/ENASC has been chartered to do. This lack of a standard to compare the ASC/ENASC responses to left the research team to attack the analysis of the ASC/ENASC responses in a different manner. The approach used was to relax the demand for a high level vision of the process to be modeled, but instead to look for a more operational definition of the process. There were two sources found for the process: the ENASC mission statement, and the responses of the customers of ASC/ENASC who had requested assistance from them. The questions were modeled after the generalized needs and customer satisfaction questions from the with and without indicator surveys, and are shown in Appendix C-4. The questionnaire responses formed a comparison between the voice of the process (in this case, ASC/ENASC) and the voice of the customer (SPO software managers and engineers who had requested assistance from ASC/ENASC recently). This comparison shows if ASC/ENASC's view of their customer's needs is the same as the need actually expressed by the customer.

The mission statement is used as a cross-check to ensure that the needs expressed in the memo are consistent with the stated objectives of the organization.

#### 3.3.2.4 Deriving Standard Responses to the ASC/ENASC Survey

Questions. The derivation for the standard responses for ASC/ENASC is done in the same manner as for the software managers and engineers. Categories of response areas are transformation, output, and voice of process. The responses are shown in Table 3.10.

TABLE 3.10

#### IDEAL SURVEY RESPONSES FOR ASC/ENASC

##### Transformation:

Question: What are the primary objectives of the database?

Ideal Response: The objective of the database is to enhance the performance of all or part of the mission of ASC/ENASC, as stated in the mission statement: systems engineering support, technical direction, computer resources acquisition policy, and standardization of avionics computer resources.

##### Output:

Question: What do you expect to provide the customer?

Ideal Response: Response should match the customer's needs as identified in their interviews.

Question: How do you expect to provide it?

Ideal Response: No standard response. Customer interviews should identify no types of products that are desired but not being provided.

##### Voice of the Process:

Question: Who do you foresee as the "customer" of the database effort?

Ideal Response: ASC program/project offices, as identified in the mission statement.

Question: What do you believe the customer needs from the database?



TABLE 3.10

IDEAL SURVEY RESPONSES FOR ASC/ENASC (CONTINUED)

Ideal Response: Response should match the customer's needs as identified in their interviews.

3.3.2.5 Outcome of Creation of the "Ideal" Software Indicator Environment. With these responses, the analysis of the survey responses can be conducted in a rigorous manner, and the amount of variation of the SPO process from the "ideal" indicator process can be determined.

3.3.2.6 Comparison of Actual Responses to Ideal Responses for the Process Flow Diagram. The second step in the analysis of the current process is the comparison of the ideal responses to the actual responses. With inputs of the process flow diagram annotated with the characteristic responses from the survey and the responses from the ideal process, the analysis of the survey responses can be conducted in a rigorous manner. The objective is to identify the problems with the ASC software indicator environment. Analysis of the SPO software manager and engineer responses was done by comparing the list of these ideal responses with the problems identified in the inputs, outputs, and transformations identified by the software managers and engineers.

Deviations from the ideal process are identified by the elimination of survey responses that identify actions or products that can be subsumed in the ideal process for a particular response area (input, output, etc).

The responses are looked at for problems in the process, examples of these type of problems might be "indicators are not used on this software development due to cost" but not "my boss ignores the indicators I show her", since the second example is a

problem with a particular program, not an endemic problem with the software indicator environment. Criterion of analysis for this area is: Indication in a survey response that there exists a problem in the ASC software indicator environment.

3.3.2.7 Outcome of the Analysis of the Current Process. The result of this analysis is a list of process problems with the current software indicator environment at ASC.

3.3.3 Comparison of Process Problems to Program Problems. The process problems may not be the only problems with the software indicator environment. There also could exist program problems, those being problems with the use of indicators already in the environment. These problems must also be considered when developing a handbook to implement software indicators. This analysis examines these problems to determine if they require consideration in the handbook. The inputs to this analysis are the process problems identified with the process flow diagram, and the program problems identified during the analysis of the responses in Section 3.2.1.

3.3.3.1 Comparison Methodology. The list of program problems is compared to the list of process problems. Underlying process problems causing program problems are identified and these program problems subsumed into the process problem.

3.3.3.2 Comparison Outcome. The remaining program problems that were not included under process problems are added to the list of problems that need to be addressed are included in a complete list of problems to be addressed in the handbook.

3.3.4 Developing Solutions to Problems. The objective of this procedure is to find answers to the process and program problems identified in the data analysis. The inputs to this analysis is the combined list of problems generated in the comparison of program and process problems, the good examples identified in the question response

analysis, and the literature available to the research team on the software indicators and the software development process.

The method for finding solutions is to first examine the problem and determine if it is in the scope of the research effort to solve. This determination is made on the basis of the research objectives. If a problem is in the scope of the research, the good examples and literature is checked to determine if there is a possible solution to the problem.

3.3.5 Outcome of the Analysis. The outcome of this analysis is a set of problems and solutions as shown in Figure 3.19. These solutions will be presented in the handbook to allow them to be implemented. These results combined with the standard set of indicators characterize the current environment and satisfy research objective number two.

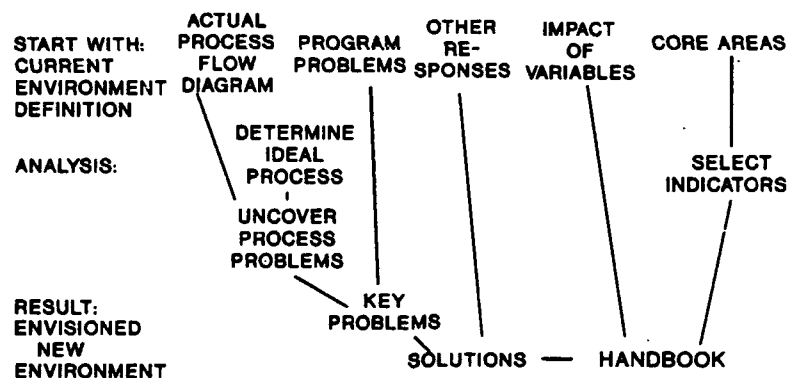


Figure 3.19 Development of the Improved Process

### 3.4 Implementation of the Improved Software Indicator Environment

The standard set of indicators and the procedures to implement them are the vehicle for implementing the improved software indicator environment. These indicators and procedures are presented in Appendix G, a handbook for the implementation of these indicators. This handbook satisfies research objective number three.

## 4.0 Analysis and Findings

### 4.1 Analysis of Question Responses.

This section discusses the results of analyzing the survey questions. Two points to remember when interpreting the data are that the averages do not always add up to 100 percent due to rounding of the results, and that in many cases the total number of respondents for a particular question may not be the same as the total number of persons interviewed. In some cases interviewees did not answer all the questions in the survey, and that changes as identified in Section 3.1.4.1.3 meant that not all questions were asked at every interview.

#### 4.1.1 Analysis of Questions Responses for Software Managers and Engineers.

This section covers an analysis of the composite results generated from the responses to the interview questions.

4.1.1.1 With Indicators Questionnaire. The composite results of the question responses for the with indicators questionnaire are shown in Table 4.1. The definition responses will be used to support the analysis of the research flow diagram and to document the current environment. The types of problems and other responses will be considered in the development of procedures for implementing the standard set of indicators in the current environment. They are discussed in the following paragraphs.

The analysis of these responses resulted in the identification of several types of problems to be addressed. The types of problems were divided into several categories. Under the category of "contractual issues" the following types of problems were identified: 1) not getting indicator data in a timely fashion, 2) the software developer

TABLE 4.1

## COMPARISON OF RESPONSES TO ACTUAL (WITH INDICATORS)

<u>Survey Question INPUTS</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
Please describe the process by which software indicator data is gathered?	<p>monthly report 35%</p> <p>telephone conversations 12%</p> <p>briefing at review 10%</p> <p>contractor forwards to SPO 10%</p>	None.	joint effort between contractor and SPO
What types of information do you currently get to help you accomplish your job?	<p>documentation 31%</p> <p>(CDRLs, reports) meetings (reviews, TIMs) 23%</p> <p>informal contact with contractor 23%</p> <p>(telecons, plant visits)</p>	None.	None.

TABLE 4.1

## COMPARISON OF RESPONSES TO ACTUAL (WITH INDICATORS) (CONTINUED)

<u>Survey Question</u>	<u>TRANSFORM</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
Please briefly describe your current job (responsibilities, functions performed, etc.)?	<b>ENGR</b>	document review 19%	None.	None.
		compare requirements to contractor performance 6%		
		software testing 12%		
		requirements 15%		
		develop/implement policy 6%		
		technical assistance 8%		
		interface w/other teams 12%		
	<b>MGR</b>			
		responsible for cost 20%		
		responsible for schedule 20%		
		interface w/other teams 10%		
		development/fielding 10%		
What happens to the information provided by the software indicators?		relay information 27%	None.	None.
		pass info/verbal updates 40%		
		trend analysis/prediction		
		identify problems, track progress		
		generate actions 29%		

TABLE 4.1  
COMPARISON OF RESPONSES TO ACTUAL (WITH INDICATORS) (CONTINUED)

<u>Survey Question OUTPUT</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
What types of information products do you produce to provide information to higher management?	verbal updates 19% activity/status report 6% program reviews 8% briefings 24% memos 8% trip reports 16%	None.	None.

TABLE 4.1

## COMPARISON OF RESPONSES TO ACTUAL (WITH INDICATORS) (CONTINUED)

<u>Survey Question</u> VOC	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
What do you believe are the primary reasons why you are currently using software indicators?	<p>Status of development effort/monitor progress 28%  insight 6%  cost purposes 10%  look for trends 6%  monitor requirements/specification 12%  compliance  monitor software performance 6%</p>	None.	None.
Have the software indicators significantly influenced Your ability to accomplish your job? Why / Why not?	<p><b>YES--74%</b>  good insight 30%  monitor spec rqmts 9%  highlight deviations from plan 9%  provide current status 17%  foresee problems 9%  <b>NO--26%</b>  information is more for contractor 33%  data too late for action 17%  can get data from other sources 17%  for higher level use 17%  indicators on contract provide information for use by management, not engineering 17%</p>	<p>All "no" responses are indicative of problems</p> <p>In addition, another problem identified was:  could have been more use if management participated and indicators contracted correctly</p>	None.



TABLE 4.1

## COMPARISON OF RESPONSES TO ACTUAL (WITH INDICATORS) (CONTINUED)

<u>Survey Question VOC</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
Have you had any changes to the software indicators you are using? Why were the changes made?	<p><b>YES--54%</b> change to better represent desired information 60% delete non-valuable indicators 27%</p> <p><b>NO--46%</b></p>	contractor refuses to provide	None.
Do you expect any future changes to the software indicators you are using?	<p><b>YES--46%</b> change indicator to better represent desired information 78% reflect changes in new requirements 11% reflect changes in supporting activity 11%</p> <p><b>NO--54%</b> late stage of program 50% program going well 40% contractor refuses 10%</p>	contractor refuses	change indicators to match program phase

TABLE 4.1

## COMPARISON OF RESPONSES TO ACTUAL (WITH INDICATORS) (CONTINUED)

<u>Survey Question VOC</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
How does this information differ from what you actually need to help you accomplish your job?	<p>information on areas 35% requirements, testing, performance</p> <p>general information on indicators 27% training, contracting</p> <p>information for better estimating 14% historical database</p>	access problems with electronic media	None.
Is there anything else you feel is important to the use of software indicators that we have not covered?	<p>gear indicators to needs/phases of program 12%</p> <p>knowledgeable people to interpret metrics 8%</p> <p>book/workshop on metrics 8%</p> <p>do not rely on indicators exclusively 8%</p> <p>more information on how to use indicators 12%</p> <p>accuracy of data important 12%</p>	None.	<p>don't just rely on indicators</p> <p>need automated tools/electronic media for indicators</p> <p>more important for contractor to use than SPO</p>

TABLE 4.1

## COMPARISON OF RESPONSES TO ACTUAL (WITH INDICATORS) (CONTINUED)

<u>Survey RESEARCH QUESTION</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
Are indicators required by contract? If no, what are the primary reasons why not? If they are on contract, how did you motivate the contractor to accept them?	<p><b>NO--40% WHY NOT?</b>  cost 18%  no emphasis/knowledge of indicators at time of contract award 55%  no influence on contractor 9%</p> <p><b>YES--60% HOW MOTIVATED</b>  contractor saw need 22%  basis for award fee 11%  "no problem" 44%  convinced contractor indicators would help track schedule/plan 11%</p>	<p>cost</p> <p>no emphasis/knowledge of indicators at time of contract award</p> <p>provide no influence on contractor</p>	<p>basis for award fee convinced contractor indicators would help track schedule/plan required by MIL-STD 1803 AFSCP 800-5 used for determining areas selected</p>
Have indicators significantly influenced the program? Why or why not?	<p><b>NO--26%</b>  more for contractor than for SPO 17%  good software development process 33%  not being used at program level 33%  get data from other sources 17%</p> <p><b>YES--74%</b>  improve visibility 25%  ID problems early 19%  helps compare actual to predicted 13%  force contractor to plan 6%  show inconsistencies 13%</p>	<p>more for contractor than for SPO</p> <p>not being used at program level</p> <p>get data from other sources</p>	<p>force contractor to plan show inconsistencies</p>

refusing to provide the data, and 3) the cost of putting indicators on contract. Under the category of "selecting the wrong indicators" the following types of problems were identified: 1) indicators not meeting the needs of the respondent by either being for a different function or for use by higher-level management, 2) indicators providing data that could have been obtained through other sources, and 3) not tailoring the indicators for specific phases of the program. Under the category of "lack of knowledge" the following types of problems were identified: 1) identifying the need for knowledgeable people to interpret metrics, and 2) a lack of knowledge about indicators themselves. The following types of problems could not be placed into a category: 1) believing that indicators are more beneficial to the contractor than the program office, and 2) a lack of management involvement.

In addition to problems, other information was obtained which could be useful in developing procedures for implementing indicators and resolving some of the problems with them. This information included having direct access to the software developer's database; tying the data provided by indicators to award fee criteria; working in joint government/contractor product teams; utilizing the ASDP 800-5, Software Development Capability/Capacity Review, to identify risk areas for supporting the selection of indicators; and MIL-STD-1803, Software Development Integrity Program, to motivate the contractor to use software indicators.

4.1.1.2 Without Indicators Questionnaire. The composite results of the question responses for the with indicators questionnaire are shown in Table 4-2. Again, the definition responses will be used to support the analysis of the research flow diagram and to document the current environment, and the types of problems will be considered in the development of procedures for implementing the standard set of indicators in the current environment.

TABLE 4.2

## COMPARISON OF RESPONSES TO ACTUAL (WITHOUT INDICATORS)

<u>Survey Question INPUTS</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
What type of information do you currently get to help you accomplish your job?	reports 17% telephone conversations 17% briefing at review 17% CDRL items 11%	None.	None.
<u>TRANSFORM</u>			
Please briefly describe your current job (responsibilities, functions performed, etc.)?	<b>ENGR</b> document review 17% software testing 9% requirements 9% technical assistance 9% interface w/other teams 17%	None.	None.
	<b>MGR</b> ECP evaluation 33% work with contractor 33% plan/schedule 33%		
How do you estimate costs, schedules, productivity, and product quality?	telephone contacts 14% reviews 18% on-line access to database 14% inputs from other teams 18% information from PM 9% data from contractor 5%	None.	on-line access to database

TABLE 4.2

## COMPARISON OF RESPONSES TO ACTUAL (WITHOUT INDICATORS) (CONTINUED)

<u>Survey Question OUTPUT</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
What types of information products do you produce to provide information to higher management?	verbal updates 8% activity/status report 33% briefings 25% memos 8% meeting minutes 8% technical papers 8%	None.	None.
<u>Survey Question VOC</u>			
What do you believe are the primary reasons why you are not currently using software indicators?	cost 22% no emphasis 22% software mature 22% end of program 11% tried indicators but failed 11%	cost no emphasis tried indicators but failed	None.
Do you envision software indicators being used on your program in the future?	too close to contract completion 14% cost 14% bad contractor relationship 14% additional software development unlikely 43% contract terminated 14%	too close to contract completion cost bad contractor relationship	None.

TABLE 4.2

## COMPARISON OF RESPONSES TO ACTUAL (WITHOUT INDICATORS) (CONTINUED)

<u>Survey Question VOC</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
How does this information differ from what you actually need to help you accomplish your job?	information on areas 60% (throughput, schedule SW problem reports) electronic data delivery 20% better, more accurate informa- tion at PMRs 20%	None.	None.
Is there anything else you feel is important to the use of software indicators that we have not covered?	contractor willingness to use indicators 13% cost of indicators 25% phase of development 13% how to define open/closed SPRs 13% tailoring of indicators to size of program 13% guidance on indicators 25%	contractor willingness to use indicators cost of indicators tailoring of indicators to size of program	None.

The types of problems identified are: 1) the costs associated with indicators, 2) the lack of emphasis placed on indicators at the time of contract award, 3) a bad relationship with the contractor which resulted in a lack of data being provided, and 4) ambiguous definitions of indicators.

4.1.1.3 Overall Findings. It is interesting to note that for the most part the with and without indicator processes are practically identical--except that one group uses indicators and the other does not. This would appear to support the belief that the software management process within ASC is fairly standardized in terms of inputs, transformation, and outputs. However, it is clearly shown that there is no standardized indicator program within the software management process as evident by the fact that 20 percent of the programs did not use indicators, 40 percent were using indicators that were not on contract, 40 percent using indicators that are on contract, and a wide range of responses *pertaining to the uses of indicators*.

In addition, it appeared as though managers and engineers that did not have indicators are still able to perform the same functions as their counterparts that did have them. This prompts the question whether software indicators are being used to obtain maximum benefit for software managers and engineers. When asked what happens to the indicator data, 27 percent of the responses were in the category of passing or relaying information, 40 percent of the responses were in the category of analyzing data, and only 29 percent of the responses were in the category of generating actions to resolve problems. None of the responses indicated that the indicator data was being used to obtain better estimates or make improvements to the process, the two actions that were identified in the literature as primary uses for indicator data.

Software indicator data can be studied in two ways, an enumerative study or an analytic study (28:184-185). An enumerative study focuses on the results or outcomes



that are observed. The analytic study focuses on what caused the outcomes or results. Based on the responses received, the implications are that software indicator data is being used for enumerative studies. Whereas some process improvements may occur from this type of analysis, it certainly is not focused on making process improvements like the analytic study.

A trend noted is that most programs did not have a separate software manager, but tended to have an engineer that did some of the functions of both manager and engineer. Four of the seven software managers in the survey were, in fact, managing both hardware and software at the same time.

In summary, the key findings of this analysis pointed to a lack of a standard process for the use of indicators, and the trend toward using the indicator data for an enumerative analyses of the software development effort, instead of in an analytic approach.

4.1.2 Analysis of Questions Responses for ASC/ENASC. The composite results of the question responses for the ASC/ENASC questionnaire are shown in Table 4-3. These responses are consistent with the results of the literature review which tied together the use of indicators and cost estimation models. The item to note is the lack of responses by the software managers and engineers concerning the use of indicator data for cost estimation, and the emphasis placed on the use of cost models. The managers and engineers believed there were problems with using indicators for the purpose of cost estimation. These problems involved obtaining a large enough database to be able to do estimations or correlations based on increasing technology and dissimilar programs. This can partially be explained by the results of the literature review which cautioned about using indicator data for dissimilar programs.

TABLE 4.3

## COMPARISON OF RESPONSES TO ACTUAL (ASC/ENASC)

<u>Survey Questions</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
<u>TRANSFORM</u>			
What are the primary objectives of the database?	independent assessment of programs surveys (software, processors, tool sets) 1750 architecture	research effort will not address 1750 architecture	None.
<u>OUTPUT</u>			
What do you expect to provide the customer?	trend analysis between programs size estimates cost estimates schedule estimates	None.	None.
How do you expect to provide it?	briefing inputs reports direct assistance	None.	None.

TABLE 4.3

## COMPARISON OF RESPONSES TO ACTUAL (ASC/ENASC) (CONTINUED)

<u>Survey Questions</u>	<u>Definition Responses</u>	<u>Problem Responses</u>	<u>Other Responses</u>
<u>Voice of the Process</u>			
Who do you foresee as the "customer" of the database effort?	program offices that need support	None.	None.
What do you believe the customer needs from the database?	software size estimates size growth estimates schedule/schedule growth estimates	None.	None.

## 4.2 Analysis of Indicators, Areas, and Publications Sheets

This section presents the results of the analysis for the interviewee-scored sheets for indicators, areas, and publications.

4.2.1 Results of Indicator Sheets Analysis. Tables 4.4 and 4.5 summarize the results of the indicator sheets analysis for managers and engineers.

TABLE 4.4

### INDICATOR ANALYSIS FOR MANAGERS' RESPONSES

<u>TYPE OF ANALYSIS</u>	<u>INDICATOR</u>	<u>CORRESPONDING AREA</u>
Pareto	Memory Utilization	Software Performance
	Throughput	Software Performance
	Requirements	Requirements
	Traceability	
Majority Responses	Requirements	Requirements
	Definition and	
	Design Stability	
	Throughput	Software Performance
	Requirements	Requirements
	Definition and	
	Design Stability	
	Memory Utilization	Software Performance
	Requirements	Requirements
	Traceability	
Weighted Average	Test Sufficiency	Testing
	Requirements	Requirements
	Compliance	
	Throughput	Software Performance
	Requirements	Requirements
	Definition and	
	Design Stability	
	Requirements	Requirements
	Traceability	
	Test Sufficiency	Testing
	Requirements	Requirements
	Compliance	
	Memory Utilization	Software Performance

TABLE 4.5  
INDICATOR ANALYSIS FOR ENGINEERS' RESPONSES

<u>TYPE OF ANALYSIS</u>	<u>INDICATOR</u>	<u>CORRESPONDING AREA</u>
Pareto	Requirements	Requirements
	Traceability	
	Requirements	Requirements
	Definition and Design Stability	
	Throughput	Software Performance
	Test Coverage	Testing
	Requirements	Requirements
	Compliance	
	Schedule Progress	Schedule
	Schedule Progress-Development and Test	Schedule
Majority Responses	Requirements	Requirements
	Traceability	
	Requirements	Requirements
	Definition and Design Stability	
Weighted Average	Throughput	Software Performance
	Requirements	Requirements
	Traceability	
	Requirements	Requirements
	Compliance	

From the results above, the analysis clearly identified Requirements to be a key area for managers and engineers. Software Performance was identified as an additional key area for managers.

4.2.2 Results of Area Sheets Analysis. Table 4.6 summarizes the results of the area sheets analysis for the managers and engineers responses.

TABLE 4.6  
AREA ANALYSIS FOR MANAGERS AND ENGINEERS

<u>TYPE OF ANALYSIS</u>	<u>MANAGERS</u>	<u>ENGINEERS</u>
Pareto	Requirements Schedule Software Performance Quality	Software Performance Requirements Testing Software Process
Majority Response	Requirements Schedule Software Performance Quality	Software Performance Requirements Testing
Weighted Average	Schedule Software Performance Quality Requirements	Requirements Software Performance

The area sheets analysis provided different information than the indicator sheets analysis. In addition to Requirements, Software Performance was also identified as a key area for engineers. This is not surprising since Software Performance was identified by two of the three types of analyses conducted on the indicator sheets. There were two additional key areas identified for managers that did not appear at all during the indicator sheet analysis: Schedule and Quality.

There was no formal analysis accomplished for the ASC/ENASC area sheets because of the small sample size. Of the two individuals who filled out the area sheets, both rated Schedule and Cost as very important with one individual also rating Manpower as very important.

4.2.3 Results of Publication Sheet Analysis. The results of the publication sheet analysis are shown in Table 4.7.

TABLE 4.7

## PUBLICATION FAMILIARIZATION

<u>PUBLICATION</u>	<u>HOW FAMILIAR</u>
AFSCP 800-43	63 percent Familiarization
IEEE STD 982	6 percent Familiarization
ESD-TR-88-01	30 percent Familiarization
AFSCP 800-14	61 percent Familiarization
SEI-12-1-1	15 percent Familiarization
AFSCP 800-49 (Draft)	18 percent Familiarization
DODD 5000.2	48 percent Familiarization
AMC 70-13	12 percent Familiarization

In general, there does not appear to be a high degree of familiarization with the indicator publications which may be a contributor to the lack of knowledge about indicators and the desire to have more information about them. This information will be compared to the familiarization respondents had with the actual indicators they rated from these publications.

4.2.4 Overall Analysis of Indicator, Area, and Publication Sheets. The responses to the indicator sheets were investigated more closely to try to obtain better insight into the differences between the indicator and area sheets responses. As discussed above, there appeared to be a strong correlation between the indicator sheet responses and the area sheet responses. A determination had to be made as to whether there was indeed a correlation between the two sheets, or were the indicator sheets actually acting as a second means to rate the software areas and the actual indicators were not being rated. This led to looking at whether the respondents were truly answering the indicator sheets

based on the knowledge of the indicators, or were they responding to the software areas that were contained in the title of the indicators. A comparison was made between the level of familiarization the respondents had with the indicators to the level of familiarization they had with the publications the indicators were derived from. This analysis is shown in Table 4.8.

TABLE 4.8

INDICATOR TO PUBLICATION FAMILIARITY CROSS-REFERENCE

<u>FAMILIAR WITH PUBLICATION</u>	<u>FAMILIAR WITH INDICATOR</u>
AFSCP 800-43: 63%	Computer Resource Utilization: 82%
	Software Development Manpower: 91%
	Cost/Schedule Deviations: 82%
	Requirements Definitions and Design Stability: 85%
	Software Progress-Development and Test: 88%
	Software Size: 91%
IEEE Std 982: 6%	Software Documentation and Source Listings: 79%
	Requirements Traceability: 94%
	Mean-Time-To-Failure: 50%
	Requirements Compliance: 79%
	Test Accuracy: 53%
ESD-TR-88-01: 30%	Schedule Progress: 97%
	Design Progress: 79%
AFSCP 800-14: 61%	Fault Density: 44%
	Test Coverage: 71%
	Testing Sufficiency: 62%
	Completeness: 68%
	Defect Density: 50%
	Documentation: 74%
AFSCP 800-49: 18% (Draft)	Memory Utilization: 91%
	Throughput: 91%
	Deviation of End Product: 29%
	Defect Distribution: 53%



Note the great disparity between the familiarization with the publication and the indicator derived from it, especially in the case of IEEE Standard 982.1. One possible cause could be that indicators from one publication are present in another. For example, Memory Utilization may also be found in AFSCP 800-43 as well as AFSCP 800-49. This is not the case because when the indicators were selected for the indicator sheet it was made certain that only unique indicators were selected from each publication.

Therefore, it was concluded that the individuals were not truly rating the indicator, but were merely rating the software area that was contained in the indicator title. The level of familiarization with software indicators that had been assumed to exist in fact did not. Because of this, the area sheets were used as the primary instrument to determine the core areas.

Table 4.9 shows the key areas for the managers, engineers, and ASC/ENASC:

TABLE 4.9		
CORE AREAS FROM AREA SHEETS ANALYSIS		
<u>MANAGERS</u>	<u>ENGINEERS</u>	<u>ASC/ENASC</u>
Requirements	Requirements	Schedule
Software Performance	Software Performance	Cost
Schedule		Manpower
Quality		

Requirements, Software Performance, and Schedule were immediately identified as core areas as they satisfied the majority criteria. Based on the functional description provided by the managers, and the fact that it did not have a majority, Quality was eliminated from being one of the core areas. This left Cost and Manpower as the last area to be decided.

Though it did not have a majority, Cost was considered to be critical for ASC/ENASC to accomplish their function. In addition, Cost was also an area that was identified by managers as part of their functional description. In discussions with ASC/ENASC, it was determined that they were really looking for data that could be used for cost estimation. This would be satisfied by knowing the level of effort applied to each software development effort. If this data is obtained for cost estimation purposes, it could also be used to support manpower estimates. For these reason, Cost was identified as a core area, with emphasis being on cost estimation, and Manpower was eliminated. However, it was determined that a level of effort is generally tied to the estimated size of the software in most cost models. Therefore, in addition to cost being identified as a core area, the need for a software size indicator was identified.

However, further investigation was warranted because of the fact that managers rated cost so low but identified it as part of their job responsibilities. The interview data could not provide sufficient information to provide an answer, but several hypotheses were developed. First, several respondents indicated that they already received cost performance reporting so a cost indicator was not necessary. Secondly, several respondents indicated that because they had a Firm Fixed Price contract cost was not an issue for them. Thirdly, software is being managed at a lower level where cost responsibility may be contained at a higher level.

The final core areas identified by the analysis were: Requirements, Software Performance, Schedule, and Cost which also requires the addition of a software size indicator. These core areas were used to determine which software indicators would be selected to comprise the standard set of indicators for ASC.

### 4.3 Analysis of Impact of Variables

The analysis of the impact of variables is divided into two main sections; answering whether the variables had an impact on the use/non-use of indicators, and also if the variables had an impact on what indicators were chosen for the programs that have indicators.

4.3.1 Impact of Variables on the Use of Indicators. An analysis was conducted of those individuals who were not using software indicators on their programs to determine if there was a consistent relationship between program variables and the use of indicators. The program variables that were evaluated were: contract type (Firm Fixed Price or Cost Plus), contract start date, contract length, type of system (embedded or mainframe application), software size (equal to or less than 100KLOC versus greater than 100KLOC), and program phase (Engineering Manufacturing Development or Production or some combination). Of the 20 programs represented by the interviews, only 4 were not using software indicators. The data for those four programs is shown in Table 4.10.

TABLE 4.10

#### VARIABLES FOR PROGRAMS NOT USING INDICATORS

<u>CONTRACT TYPE</u>	<u>START DATE</u>	<u>LENGTH (Years)</u>	<u>SYSTEM TYPE</u>	<u>SIZE (KLOC)</u>	<u>PHASE</u>
Firm Fixed Price	88	5	Embedded	Unknown	Modification
Firm Fixed Price	84	10	Embedded	Unknown	Production
Firm Fixed Price	88	4	Mainframe	70	Combined
Cost Plus	87	5	Mainframe	100	Combined

Based on the small sample size and the apparent inconsistency of the data, no conclusions could be drawn regarding the variables and the use of indicators. The only variable that showed some consistency was contract type with 3 of the 4 programs being Firm Fixed Price. However, there also were eight Firm Fixed Price programs that were also using indicators.

To provide further insight into any relationship between the variables and the use of indicators, responses to the interview question, "what do you believe are the primary reasons why indicators are not being used on your program?", were analyzed. The responses revealed three main reasons for not using indicators: cost, a lack of emphasis at time of contract award, or the software was too mature. From these responses, it was concluded that there was no apparent trend between the variables and their impact on whether or not software indicators were used. Instead, it would appear that there are different variables that have a greater impact on the use of indicators such as cost of using indicators and development phase of the program.

4.3.2 Impact of Variables on Needs. The results of this analysis are shown in Figures 4.1 and 4.2. From these results, it appeared that there were several elements that had an effect on the need for the core areas. However, when investigating closer, it was determined that the sample size was too small in many cases to be able to have sufficient information to come to any conclusion. For example, in the elements of program size and type of application, it appeared as though there was a significant difference. However, these were sample sizes of 5 and 7 respectively with 3 respondents being in both groups. With such small sample sizes, one individual can have a significant impact. Therefore, based on the available data, it cannot be ascertained whether these elements do have an impact on the needs for indicators or whether a few individuals are skewing the analysis results. Though no conclusions could be reached, this information

may provide some hypotheses for future research efforts to study.

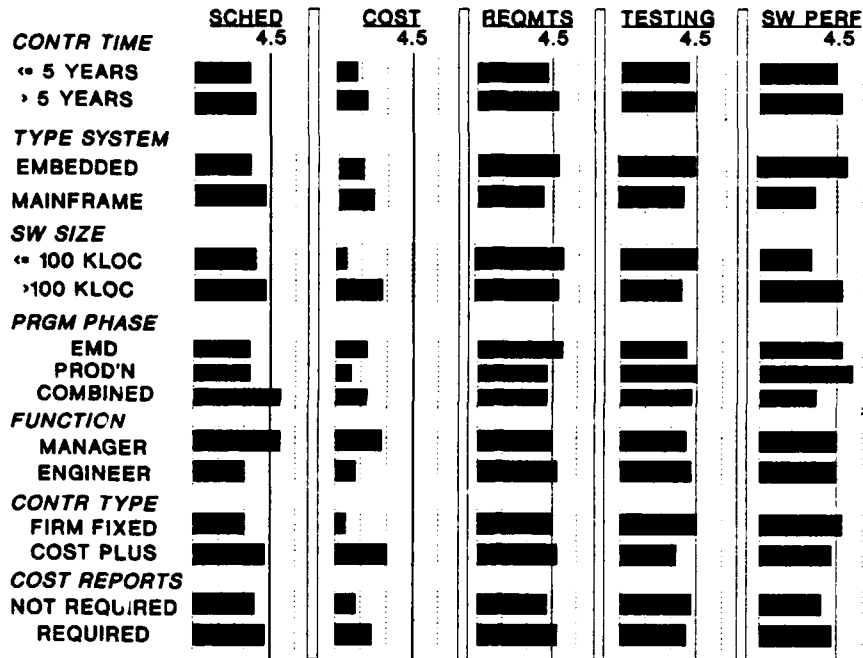


Figure 4.1 Weighted Average Analysis of Impact of Variables

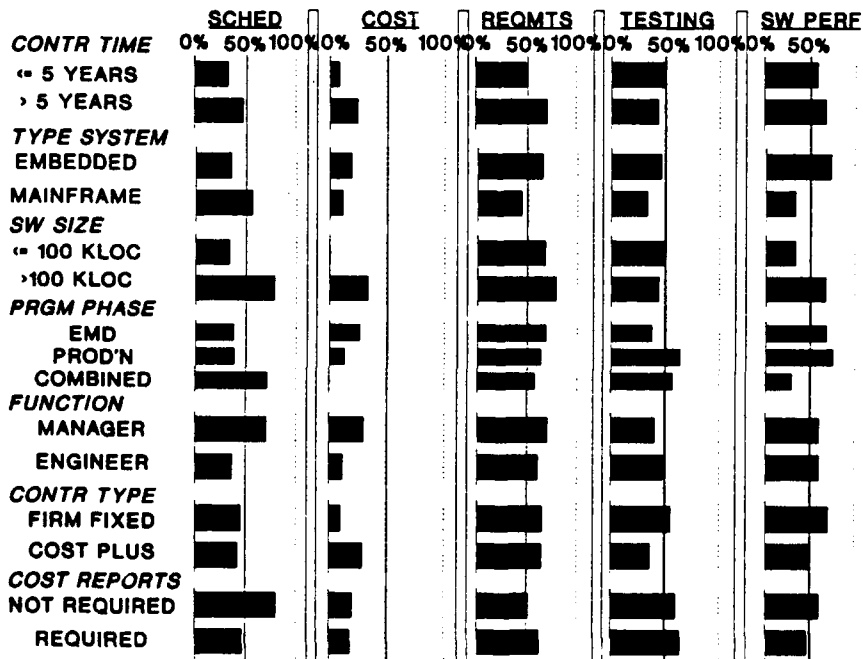


Figure 4.2 Majority Response Analysis of Impact of Variables

#### 4.4 Analysis of the Research Flow Diagram.

The analysis of the research flow diagram surfaced major flaws with the way it represents the current environment. The primary flaw is in the way it represents the relationship between the software managers/engineers and ASC/ENASC. The research flow diagram shows the current environment to be a serial process from beginning to end. In reality there are two distinct processes that make up the current environment. The first is the process that involves software managers and engineers responsible for the software development effort. The second is the process that involves ASC/ENASC providing independent schedule and cost estimates to software managers and engineers. This is evident by the differences in the responses when software managers and engineers were given the with/without indicators questionnaires and when they were given the voice of the customer questions from the ASC/ENASC questionnaire. When given the with/without indicators questionnaires, cost was not rated highly by either managers or engineers and there was no mention of obtaining independent estimates from ASC/ENASC as additional sources of information. However, when asked the voice of the customer portion of the ASC/ENASC questionnaire, not only did cost and schedule estimates make up 75 percent of the responses, but the respondents unanimously agreed that they would utilize the services of ASC/ENASC again. There is an interrelationship that exists between the managers/engineers and ASC/ENASC represented by the managers and engineers supplying indicator data to ASC/ENASC who in turn supplies independent estimates of schedule and cost. The final process flow diagram is shown in Figure 4.3.

This final process flow diagram combined with the responses from the questionnaires documents the current environment and the vision of the improved process can now be created.

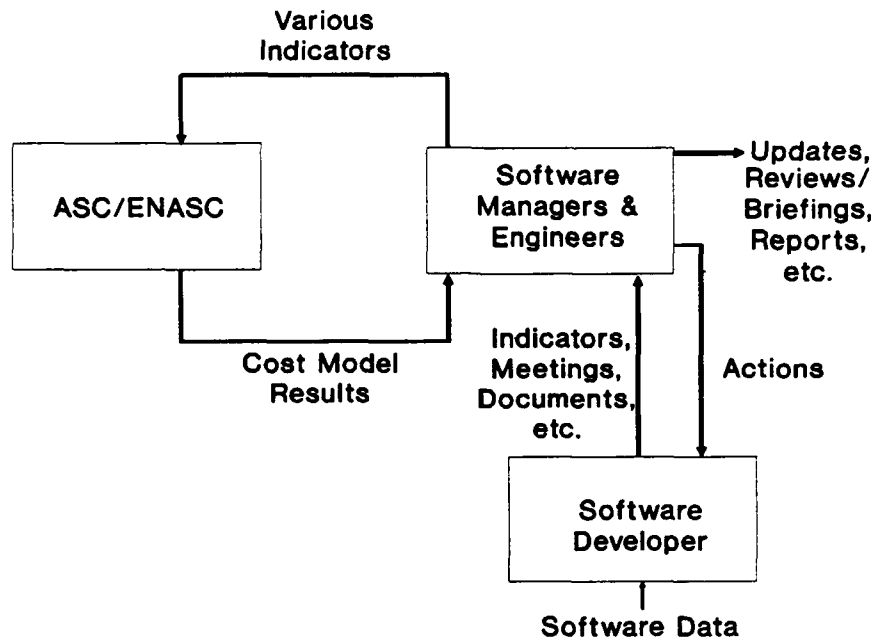


Figure 4.3 Final Flow Diagram

#### 4.5 Selection of Standard Indicators.

The method for selecting the standard set of indicators has four parts: 1) gathering a list of all indicators that apply to the core areas identified, 2) selecting an indicator for each area, 3) considering optimization of the indicator that may be needed, and 4) developing procedures based on the source of the indicator for its implementation.

The selection of indicators from the database was done by sorting on the PRIME\_USE field. Publications that the indicators are sourced out of are: 1) AFSCP 800-43, Software Management Indicators, both the 31 August 1990 version and the 31 Jan 1986 version; 2) SQI, a report on Software Quality Indicators from Scientific Systems Inc, 24 Sep 1986; 3) ESD-TR 88-01, an ESD technical report on Software Management Metrics, May 1988; 4) SEI 12-1, the Software Engineering Institute curriculum module on Software Metrics. SEI-CM-12-1.1, December 1988; and 5) AFSCP 800-49, Software

Quality Measurement: The Product and the Process (draft). Table 4.11 shows each area, the indicator titles of the indicators in that area given to us by a software manager or engineer, and the indicators from the database that provide information in the same area. The table also provides a short comparison of the relative merit of the database indicator to the indicator from a given program. In table 4.11, "publication" refers to the published indicator from the database, and "indicator" refers to the indicator provided by the respondent. It should be noted that there were no program indicators on cost. Therefore, the area of cost is not represented in Table 4.11.

TABLE 4.11  
INDICATOR SOURCES AND COMPARISON

SCHEDULE INDICATORS

Program Indicator: CSCI REQUIREMENTS DEVELOPMENT STATUS

AFSCP 800-43 (1990) CSCI DESIGN PROGRESS  
Publication does not track requirements allocation, starts with generic CSU's designed total for each CSCI.

AFSCP 800-43 (1986) SOFTWARE PROGRESS, DEVELOPMENT AND TEST  
Tracks to overall CSCI design, not to CSU's of a CSCI, does not indicate replanning.

Program Indicator: CSCI FQT TEST PROCEDURES DRY RUN STATUS

AFSCP 800-43 (1990) FQT PROGRESS  
Publication tracks identical to indicator.

Program Indicator: CSCI CSC TEST STATUS

Same as above indicator, but scale is CSU's vice tests.

Program Indicator: CSCI CODE AND UNIT TEST STATUS

AFSCP 800-43 (1990) CSCI CODE AND UNIT TEST  
Publication tracks to indicator, but includes detail as to redesign effort that indicator is missing



TABLE 4.11 (continued)

INDICATOR SOURCES AND COMPARISON

SCHEDULE INDICATORS

Program Indicator: CSCI CODE AND UNIT TEST STATUS (continued)

AFSCP 800-43 (1986) SOFTWARE PROGRESS, DEVELOPMENT AND TEST  
Data is integrated into graph with design and integration detail.

Program Indicator: CSCI DETAILED DESIGN STATUS

AFSCP 800-43 (1990) CSCI DESIGN PROGRESS  
Publication tracks to indicator, but includes redesign details.

AFSCP 800-43 (1986) SOFTWARE PROGRESS, DEVELOPMENT AND TEST  
Publication tracks indicator, shows combined graph with test and integration status.

Program Indicator: CSCI PRELIMINARY DESIGN STATUS

Apparently identical data to CSCI requirements development status, perhaps considering preliminary design vs. requirements allocation. No publication indicator treats this separately.

Program Indicator: STATUS INDICATORS DESIGN PROGRESS

AFSCP 800-43 (1990) CSCI DESIGN PROGRESS  
Publication tracks to indicator, but indicator includes walkthrough milestone.

AFSCP 800-43 (1986) SOFTWARE PROGRESS, DEVELOPMENT AND TEST  
Publication does not explicitly talk to SCU (units) and does not cover walkthroughs.

Program Indicator: CODE AND INTEGRATION STATUS

AFSCP 800-43 (1990) CSCI INTEGRATION AND TEST PROGRESS  
Publication covers indicator and includes details on recode efforts.

AFSCP 800-43 (1986) SOFTWARE PROGRESS, DEVELOPMENT AND TEST  
Portion of publication indicator completely covers indicator.

TABLE 4.11 (continued)

INDICATOR SOURCES AND COMPARISON

SCHEDULE INDICATORS

Program Indicator: DESIGN VERIFICATION TEST

No publication covers a test of this name

AFSCP 800-43 (1990)

Discusses FQT progress, including total tests for a CSCI and their schedule. This might cover DVT's.

Program Indicator: PROGRESS PERFORMANCE METRIC

Actual tasks specified for indicator unstated. Generic metric applicable to painting battleships as well as designing software.

Program Indicator: CSCI PRODUCT COMPLETION STATUS

Indicator tracks KSLOC completed of a CSCI. All publications track by CSU.

Program Indicator: OFP STATUS

Indicator tracks requirements allocation units designed, coded and tested, and integration.

AFSCP 800-43 (1986) SOFTWARE PROGRESS, DEVELOPMENT AND TEST

Publication does not cover requirements allocation, but does cover unit design, code and test, and integration.

AFSCP 800-43 (1990) CSCI DESIGN PROGRESS / CSCI CODE AND UNIT  
TEST PROGRESS /CSCI INTEGRATION AND TEST PROGRESS

Publication design progress indicator covers unit design, code and test covers code and testing CSCI integration and test progress provides additional integration test detail.

Program Indicator: CODING STATUS

AFSCP 800-43 (1990) CODE AND UNIT TEST PROGRESS

Publication tracks to indicator, but includes test details.

AFSCP 800-43 (1986) SOFTWARE PROGRESS, DEVELOPMENT AND TEST

Indicator covers percent code completed, but includes many additional details.

TABLE 4.11 (continued)

INDICATOR SOURCES AND COMPARISON

SCHEDULE INDICATORS

Program Indicator: DESIGN STATUS

AFSCP 800-43 (1990) CSCI DESIGN PROGRESS

Publication tracks to indicator, but includes explicit redesign details.

AFSCP 800-43 (1986) SOFTWARE PROGRESS, DEVELOPMENT AND TEST

Indicator covers percent design completed, but includes many additional details.

Program Indicator: CODING AND UNIT TEST STATUS

AFSCP 800-43 (1986) SOFTWARE PROGRESS, DEVELOPMENT AND TEST

Indicator covers percent of coding, unit test status completed, but includes many additional details.

AFSCP 800-43 (1990) CSCI CODE AND UNIT TEST

Publication tracks to indicator, but provides additional redesign information.

Program Indicator: REQUIREMENTS ALLOCATION STATUS

No publication covers requirements allocation

AFSCP 800-43 (1990) has requirements stability indicator.

Program Indicator: PROCEDURE DEVELOPMENT AND TEST

No publication talks to procedures, but several talk to CSCI's and CSU's.

Program Indicator: ESTIMATED TEST CONDUCT COMPLETION

Indicator covers number of test runs.

AFSCP 800-43 (1990) FQT TEST PROGRESS

Publication covers FQT tests completed, indicator covers test runs completed, not necessarily tests completed.

Program Indicator: DEVELOPMENT MILESTONES

Indicator milestones are tracked planned/actuals/replan/delta

No publication tracks milestones as such.

TABLE 4.11 (continued)

INDICATOR SOURCES AND COMPARISON

REQUIREMENTS

Program Indicator: CSCI IDD DESIGN CHANGE HISTORY

AFSCP 800-43 (1990) CSCI IRS AND IDD PROGRESS

Publication stresses total IDD requirements, actual, planned and revised - indicator stresses changes, adds/deletes, and the cumulative total of changes.

AFSCP 800-49 (DRAFT) SOFTWARE REQUIREMENTS STABILITY

Publication deals generically with documents, tracks change rate as it approaches 0.

Program Indicator: SQI SYSTEM REQUIREMENTS STABILITY

Publication talks of SSS and OCD, and tracking of approved changes.

Program Indicator: CSCI REQUIREMENTS CHANGE HISTORY

AFSCP 800-49 (DRAFT) SOFTWARE REQUIREMENTS STABILITY

Publication deals generically with documents, tracks change rate as it approaches 0.

SQI SYSTEM REQUIREMENTS STABILITY

Publication talks of SSS and OCD, and tracking of approved changes.

Program Indicator: CSCI SDD DESIGN CHANGE HISTORY

AFSCP 800-43 (1990) CSCI SRS AND SDD PROGRESS

Publication tracks numbers (revised) of requirements to be documented in SDD, indicator stresses only changes.

AFSCP 800-49 (DRAFT) SOFTWARE REQUIREMENTS STABILITY

Publication deals generically with documents, tracks change rate as it approaches 0.

SQI SYSTEM REQUIREMENTS STABILITY

Publication talks of SSS and OCD, and tracking of approved changes.

Program Indicator: CSCI SRS REQUIREMENTS CHANGE HISTORY

AFSCP 800-43 (1990) CSCI SRS AND SDD PROGRESS

Publication matches close to indicator, but indicator stresses changes

TABLE 4.11 (continued)

INDICATOR SOURCES AND COMPARISON

REQUIREMENTS

Program Indicator: REQUIREMENTS/DESIGN STABILITY METRIC

AFSCP 800-43 (1990) CSCI REQUIREMENTS STABILITY/CSCI DESIGN STABILITY

Publication tracks completely to indicator.

AFSCP 800-49 (Draft)

Publication deals with problems found, like other changes.

SEI 12-1

Publication speaks in general terms to design changes and tracking them

SIZE

Program Indicator: SOFTWARE SIZE STABILITY METRIC

AFSCP 800-43 (1990) SOFTWARE SIZE

Publication covers primitives of stability but does not explicitly stress stability.

ESD-TR 88-01 SOFTWARE SIZE

Publication has same primitives, explains different reasons for increasing/decreasing size.

Program Indicator: SOFTWARE SIZE

AFSCP 800-43 (1990) SOFTWARE SIZE

Publication covers same area as indicator, but does not include percent assembler.

ESD-TR 88-01 SOFTWARE SIZE

Publication covers same area as indicator, but does not include percent assembler.

Program Indicator: CSCI SIZE STATUS

AFSCP 800-43 (1990) SOFTWARE SIZE

Publication same as indicator, but does not track estimated reusable or estimated modified.

TABLE 4.11 (continued)

INDICATOR SOURCES AND COMPARISON

SIZE

Program Indicator: CSCI SIZE STATUS (continued)

ESD-TR 88-01 SOFTWARE SIZE

Publication same as indicator, but does not track estimated reusable or estimated modified.

SOFTWARE PERFORMANCE

Program Indicator: I/O BUS THROUGHPUT CAPACITY

AFSCP 800-43 (1990) I/O UTILIZATION

Publication same as indicator, but indicator defines "Management Reserve" and "Effective bus capacity".

AFSCP 800-43 (1986) COMPUTER RESOURCE UTILIZATION

Publication contains I/O use indicator, but indicator defines "Management Reserve" and "Effective bus capacity".

Program Indicator: PROCESSOR MEMORY UTILIZATION

AFSCP 800-43 (1990) MEMORY UTILIZATION

Publication same as indicator, but indicator defines "Management Reserve" and "Effective bus capacity".

AFSCP 800-43 (1986) COMPUTER RESOURCE UTILIZATION

Publication contains degree of memory use, but indicator defines "Management Reserve" and "Effective bus capacity".

Program Indicator: PROCESS MEMORY UTILIZATION FOR A CSCI

No publications break memory use down to individual CSCI's.

Program Indicator: PROCESSOR THROUGHPUT UTILIZATION

AFSCP 800-43 (1986) COMPUTER RESOURCE UTILIZATION

Publication matches indicator, but does not break use down for CSCI's.

AFSCP 800-43 (1990) CPU UTILIZATION

Publication matches indicator, but does not break use down for CSCI's.

TABLE 4.11 (continued)

INDICATOR SOURCES AND COMPARISON

SOFTWARE PERFORMANCE

Program Indicator: PROCESSOR THROUGHPUT UTILIZATION FOR A CSCI  
No publications break throughput down to individual CSCI level.

Program Indicator: COMPUTER RESOURCE UTILIZATION

AFSCP 800-43 (1990) COMPUTER RESOURCE UTILIZATION  
Publication tracks to indicator - indicator contains detail limits for reserves.

Program Indicator: RESOURCE UTILIZATION METRIC

AFSCP 800-43 (1990 and 1986) COMPUTER RESOURCE UTILIZATION  
Both publications track to indicator; indicator mandates use through testing phase.

Program Indicator: OFP RAM/EPROM USAGE

AFSCP 800-43 (1990 and 1986) MEMORY UTILIZATION  
Both publications track to indicator, but do not differentiate EEPROMS and RAM.

The following section will detail the selection of an indicator for each core area.

4.5.1 Selection of Schedule Indicator Schedule indicators were divided into several subclasses as both the literature and the programs that provided them showed the variations that were possible. The basic differences were that the schedule indicators tracked a particular phase of the software: code status, test status, code/test status, design status, requirements analysis status, integration status, and overall status. Some references for indicators had various phases combined, providing a view of the interrelationships between phases, but it was determined that these presentations were hard to decipher, relatively inflexible, and noted that the majority of the programs using

these indicators did not combine them. For these reasons, the following indicators were selected: Resource Allocation Status, Preliminary Design Status, Detailed Design Status, Code and Unit Test Status, and Integration Status. These indicators are based upon indicators found in AFSCP 800-43 with the modification of adding a percent difference between planned and actual activities for each reporting period. This addition allows for examination of variability from period to period with the expectations that this will improve problem recognition by establishing upper and lower control limits.

4.5.2 Selection of a Cost Indicator. The only indicator found that addressed the cost area was Cost/Schedule Deviations from AFSCP 800-43. This indicator does not address the level of effort but is based on the Cost/Schedule Control Systems Criteria displaying Budgeted Cost of Work Performed, Budgeted Cost of Work Completed, and Actual Cost of Work Performed. Therefore, no cost indicators were selected. However, based on the need to obtain information for cost estimation purposes, a level of effort indicator is provided for validation. This indicator is a modification of the status indicators selected in the previous section with man months being the primitive for the indicator. This indicator will be included in the draft handbook provided in Appendix G, but again it should be noted that this is only a recommendation and the indicator should be validated before it is used.

4.5.2.1 Selection of a Size Indicator. There were several versions of size indicators in the programs that provided indicators. Again, the presentations in these indicators were at the extremes, either very complex or very generic. The best presentation was in AFSCP 800-43 (1990). The tracking of size estimates should be done separately from the actual data to reduce the complexity of the presentation as displaying new, modified, and reused code in both actual and estimate form on one graph makes it very busy.



4.5.3 Selection of a Requirements Indicator. As noted in table 4.11, the indicators being used by the programs were very similar to the publications. The two indicators that present the best picture of the requirements are in AFSCP 800-43, CSCI Requirements Stability and CSCI Design Stability. There are indicators in IEEE Standard 982.1 that can provide very detailed information concerning the tracking of requirements through the software development phases and ensuring they are met. However, these indicators are very complex, require large amounts of data and would not be cost-effective. For these reasons, these indicators were not selected.

4.5.4 Selection of Software Performance Indicators. All the program computer resource utilization indicators tracked the three areas identified in AFSCP 800-43, throughput, memory use, and input/output use. The results of the indicator sheets analysis clearly showed that software managers and engineers preferred the throughput and memory indicators over the computer resource utilization indicator (majority responses analysis shows throughput with 55%, memory with 42%, and computer resource utilization with 29%). These results, combined with most programs having the performance also broken out, resulted in the selection of the following indicators: I/O Bus Throughput Capability, Processor Memory Utilization, and Processor Throughput Utilization. These indicators track very closely to AFSCP 800-43 with tracking of the indicator data at the CSCI level.

#### 4.6 The Standard Set of Indicators with Recommendations for Use.

This section presents the standard set of indicators that were selected with recommendations for using them.

4.6.1 Schedule Indicators. There are six indicators within the schedule area are: Requirements Allocation Status, Preliminary Design Status, Detailed Design Status, Code and Unit Test Status, and Integration Status.

4.6.1.1 Resource Allocation Status. The purpose of the Resource Allocation Status indicator is to track the progress of allocating requirements to the software CSCIs through the development of the Software Requirements Specification (SRS) and Interface Requirements Specification (IRS).

4.6.1.1.2 Frequency of Update. This indicator will be updated monthly from the System Requirements Review until approval of the Software Requirements Specifications (SRS) and Interface Requirements Specifications (IRS).

4.6.1.1.3 Update Criteria. Data for updates will be obtained by weekly inspections of the completed portions of the SRS and IRS.

4.6.1.1.4 Indicator Inputs. The indicator is constructed from the following information:

a. Number of Requirements Allocated to the CSCIs: Data is obtained from the requirements in the System/Segment Design Document to be allocated to the CSCI.

b. Planned Number of Requirements Allocated: Data is obtained from estimated number of requirements each reporting period expected to complete requirements analysis and be incorporated in the SRS and IRS.

c. Actual Number of Requirements Allocated: Data is obtained from actual number of requirements documented in the SRS IRS for each reporting period.

d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations (18:324).

4.6.1.1.5 An Example of the Indicator. An example of the Requirements Allocation Status indicator is shown in Figure 4.4.

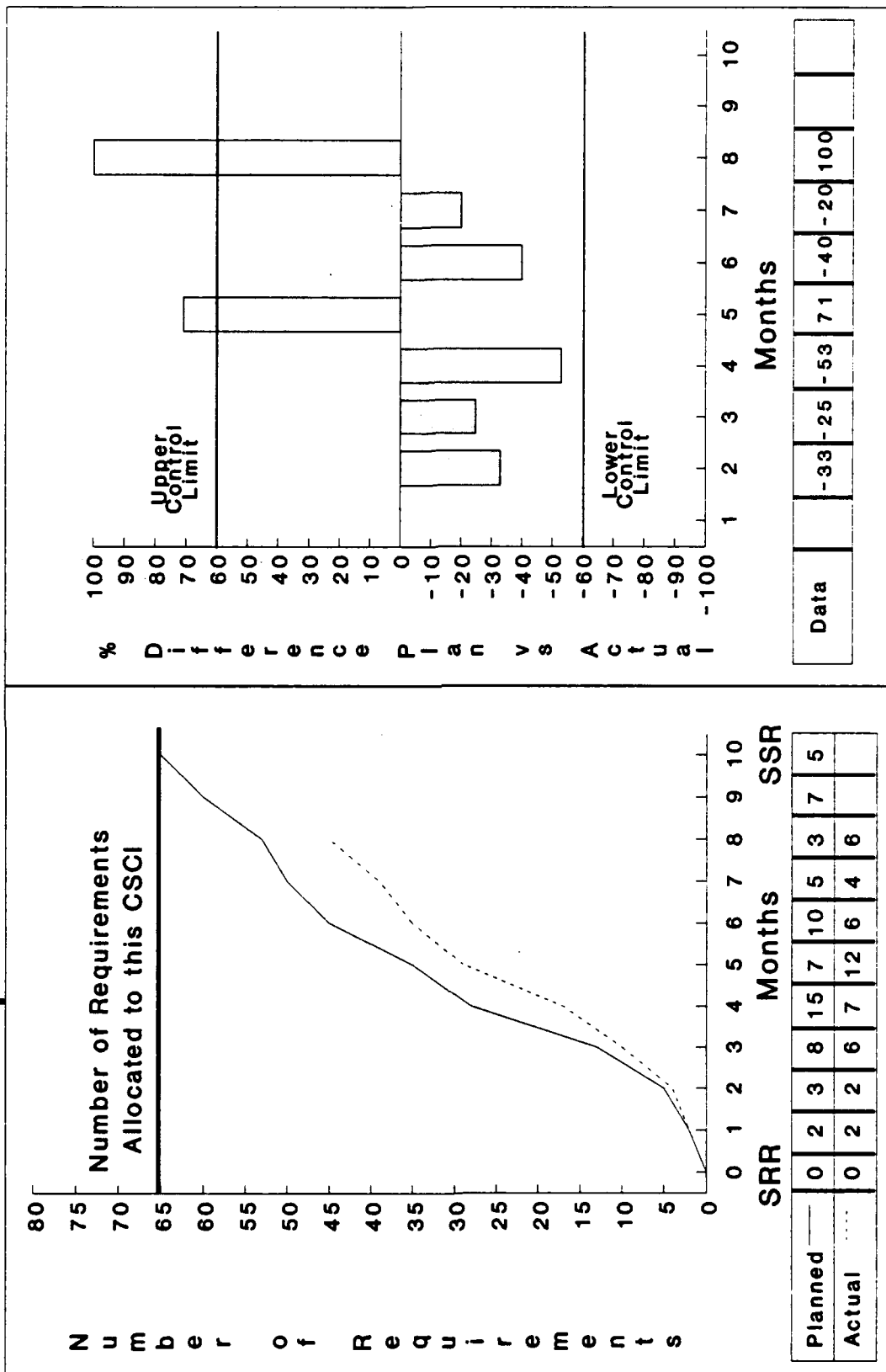
4.6.1.1.6 Using the Indicator. The indicator provides information concerning how well the resource allocation activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the resource allocation effort while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the resource allocation task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.6.1.1.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Have separate indicators for the SRS and IRS
- b. Breakout Planned and Actual data for specific sections of the SRS and IRS (Sections 3 and 4).
- c. Change frequency of updates to meet your needs.
- d. Change time period of indicator to meet your needs.

4.6.1.2 Preliminary Design Status. The purpose of the Preliminary Design Status indicator is to track the progress of translating the requirements into design for the

# CSCI (Name of CSCI) Requirements Allocation Status



As of:

software CSCs through the development of the Software Design Document (SDD) and Interface Design Document (IDD).

4.6.1.2.1 Frequency of Update. This indicator will be updated monthly from the Software Specification Review until the Preliminary Design Review.

4.6.1.2.2 Update Criteria. Data for updates will be obtained by weekly inspections of the completed portions of the Software Design Document (SDD) and the Interface Design Document (IDD).

4.6.1.2.3 Indicator Inputs. The indicator is constructed from the following information:

a. Number of Requirements Allocated to the CSCIs: Data is obtained from the number of requirements in the SRS for the CSCI.

b. Planned Number of Designs Documented: Data is obtained from the estimated number of requirements each reporting period expected to complete preliminary design and be incorporated into the SDD and IDD.

c. Actual Number of Designs Documented: Data is obtained from actual number of requirements documented in the SDD and IDD for each reporting period.

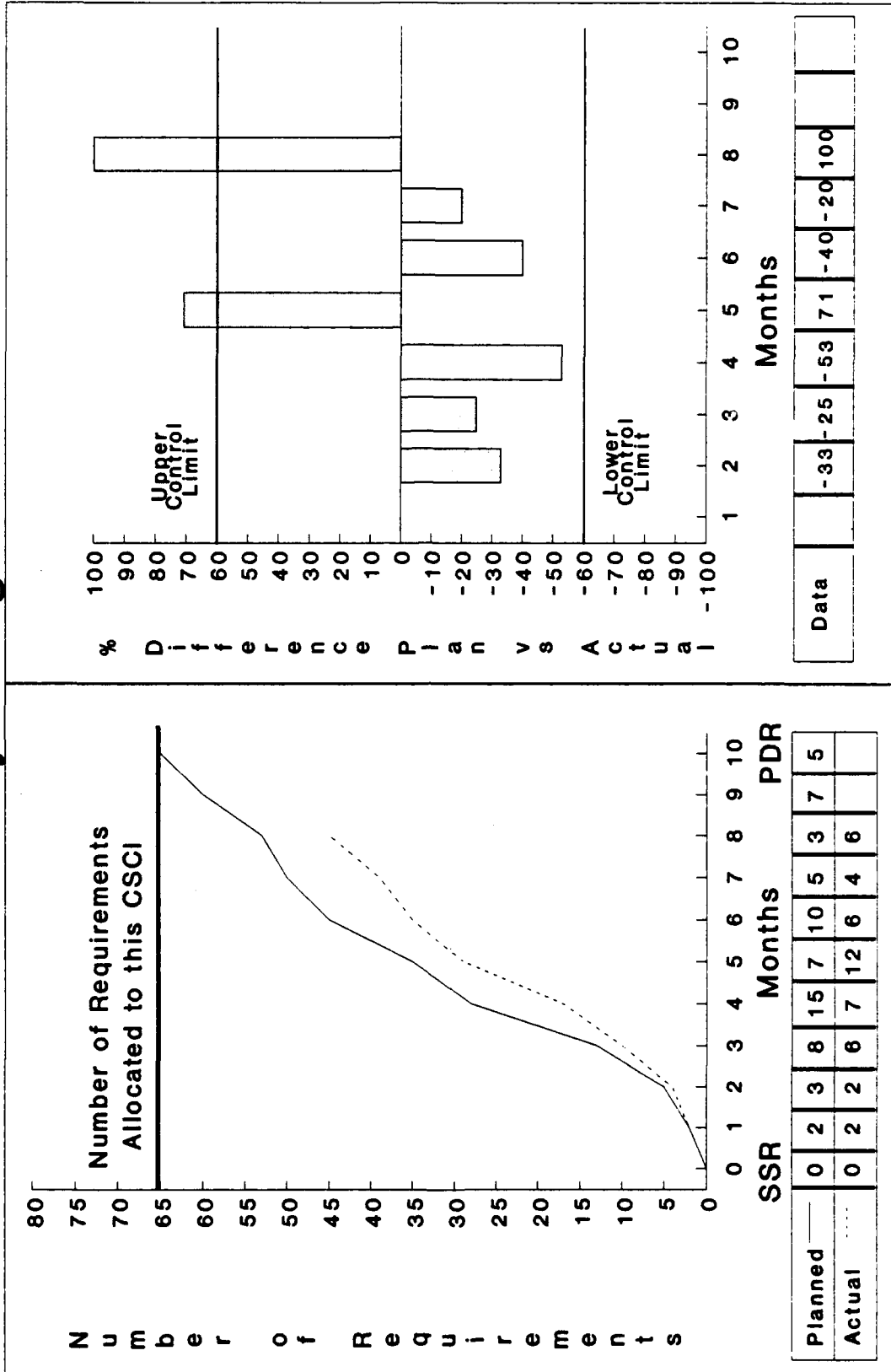
d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations (18:324).

4.6.1.2.4 An Example of the Indicator. An example of the Preliminary Design Status indicator is shown in Figure 4.5.

# CSCI (Name of CSCI) Preliminary Design Status



As of:

4.6.1.2.5 Using the Indicator. The indicator provides information concerning how well the preliminary design activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the preliminary design effort while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the preliminary design task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.6.1.2.6 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Have separate indicators for the SDD and IDD
- b. Breakout Planned and Actual data for specific sections of the SDD and IDD (Sections 3 and 4).
- c. Change frequency of updates to meet your needs.
- d. Change time period of indicator to meet your needs.

4.6.1.3 Detailed Design Status The purpose of the Detailed Design Status indicator is to track the progress of translating the requirements into design for the software CSUs.

4.6.1.3.1 Frequency of Update. This indicator will be updated monthly from the Preliminary Design Review until the Critical Design Review.

4.6.1.3.2 Update Criteria. Data for updates will be obtained by weekly inspections to determine the completeness of the detailed design for the CSUs.

4.6.1.3.3 Indicator Inputs. The indicator is constructed from the following information:

- a. Total Number of CSUs in the CSCI: Data is obtained from estimates or actual numbers if known.
- b. Planned Number CSUs to Complete Detailed Design: Data is obtained from the estimated number of CSUs each reporting period that will complete detailed design and be documented in the SDD and IDD.
- c. Actual Number of Requirements Allocated: Data is obtained from actual number of CSUs completing detailed design and documented in the SDD and IDD for each reporting period.
- d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

- e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations (18:324).

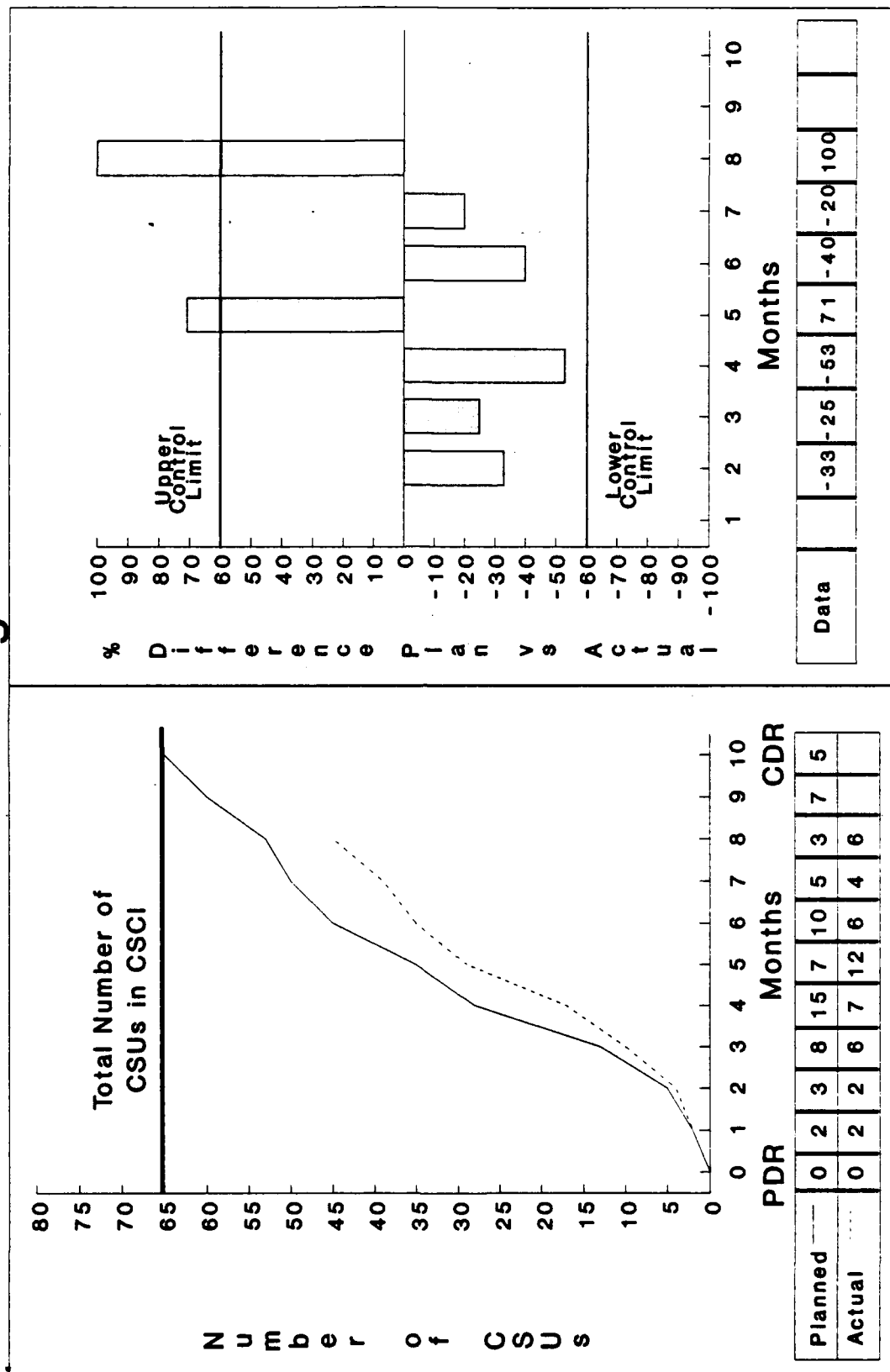
4.6.1.3.4 An Example of the Indicator. An example of the Detailed Design Status indicator is shown in Figure 4.6.

4.6.1.3.5 Using the Indicator. The indicator provides information concerning how well the detailed design activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the detailed design effort while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the detailed design task may be getting back on track, but the bottom graph shows large deviations between the planned and actual



# CSCI (Name of CSCI) Detailed Design Status

Figure 4.6 Example of Detailed Design Status Indicator



As of:

activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.6.1.3.6 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Have separate indicators for the SDD and IDD
- b. Breakout Planned and Actual data for specific sections of the SDD and IDD (Sections 3 and 4).
- c. Change frequency of updates to meet your needs.
- d. Change time period of indicator to meet your needs.

4.6.1.4 Code and Unit Test Status. The purpose of the Code and Unit Test Status indicator is to track the progress of coding and testing the CSUs.

4.6.1.4.1 Frequency of Update. This indicator will be updated monthly from the Critical Design Review until the Test Readiness Review has been successfully completed.

4.6.1.4.2 Update Criteria. Data for updates will be obtained by weekly inspections to determine the number of CSUs that have been coded AND successfully passed testing.

4.6.1.4.3 Indicator Inputs. The indicator is constructed from the following information:

- a. Total Number of CSUs in the CSCI: Data is obtained from estimates or actual numbers if known.

b. Planned Number of CSUs to Complete Code and Testing: Data is obtained from the estimated number of CSUs each reporting period that will complete coding and testing.

c. Actual Number of CSUs to Complete Code and: Data is obtained from actual number of CSUs completing coding and testing for each reporting period.

d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

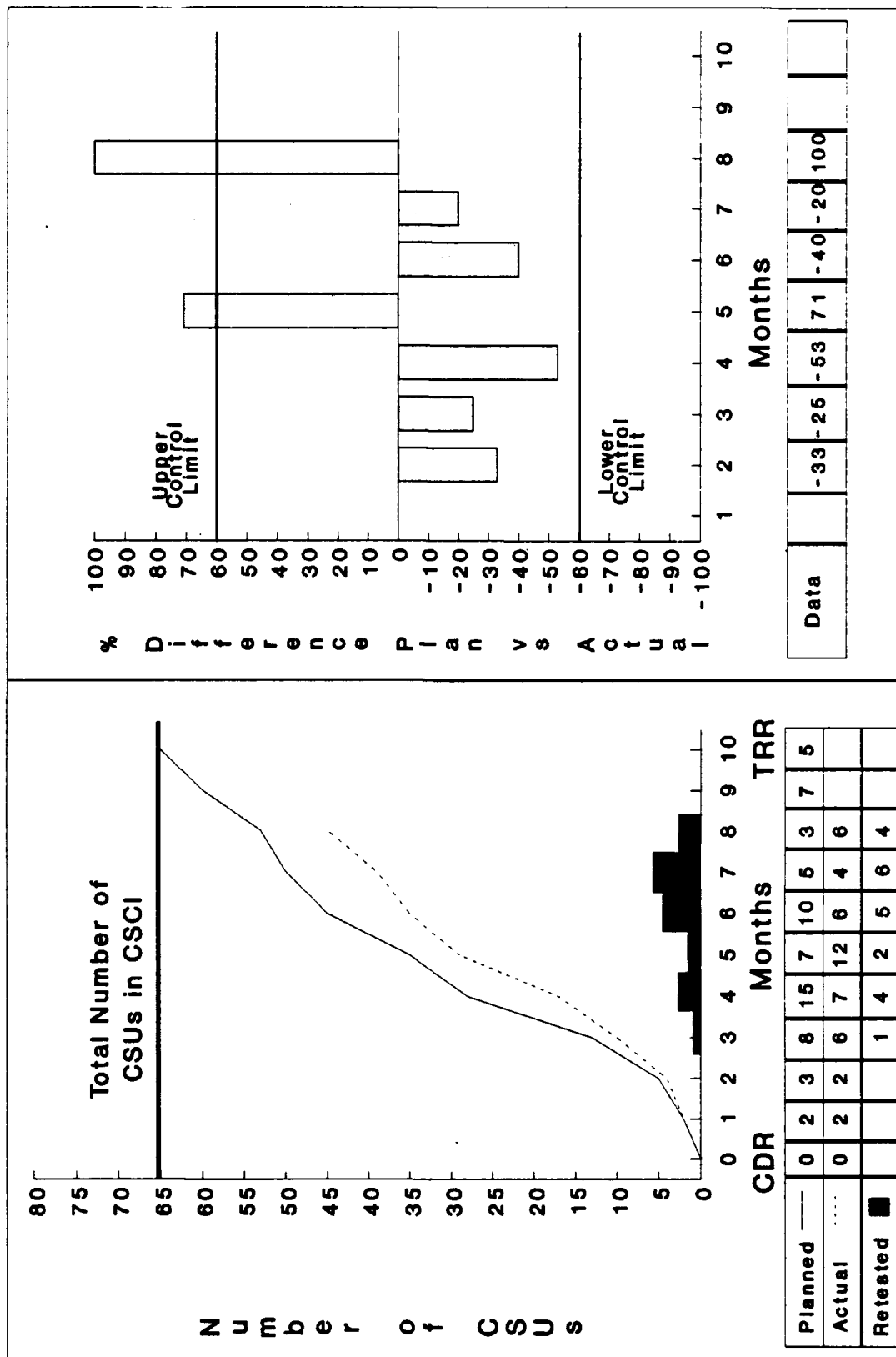
$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations (18:324).

4.6.1.4.4 An Example of the Indicator. An example of the Code and Unit Test Status indicator is shown in Figure 4.7.

4.6.1.4.5 Using the Indicator. The indicator provides information concerning how well the coding and testing activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the coding and testing effort while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the coding and testing task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

# CSCI (Name of CSCI) Code and Unit Test Status



As of:

4.6.1.4.6 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Change time period of indicator to meet your needs.

4.6.1.5 Integration Status The purpose of the Integration Status indicator is to track the progress of integrating the CSUs into the CSCI.

4.6.1.5.1 Frequency of Update. This indicator will be updated monthly from the Test Readiness Review until integration has been successfully completed.

4.6.1.5.2 Update Criteria. Data for updates will be obtained by weekly inspections to determine the number of CSUs that have been successfully integrated into a CSCI. This requires the integration of a group of CSUs, usually comprising a CSC, and passing testing requirements for the performance of the CSCI.

4.6.1.5.3 Indicator Inputs. The indicator is constructed from the following information:

- a. Total Number of CSUs in the CSCI: Data is obtained from estimates or actual numbers if known.
- b. Planned Number of CSUs to Complete Integration: Data is obtained from the estimated number of CSUs each reporting period that will complete integration.
- c. Actual Number of CSUs to Complete Integration: Data is obtained from actual number of CSUs completing integration for each reporting period.
- d. Number of CSUs Undergoing Retesting: Data is obtained from the actual number of CSUs that have failed integration, been recoded, and are being retested.
- e. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

f. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations (18:324).

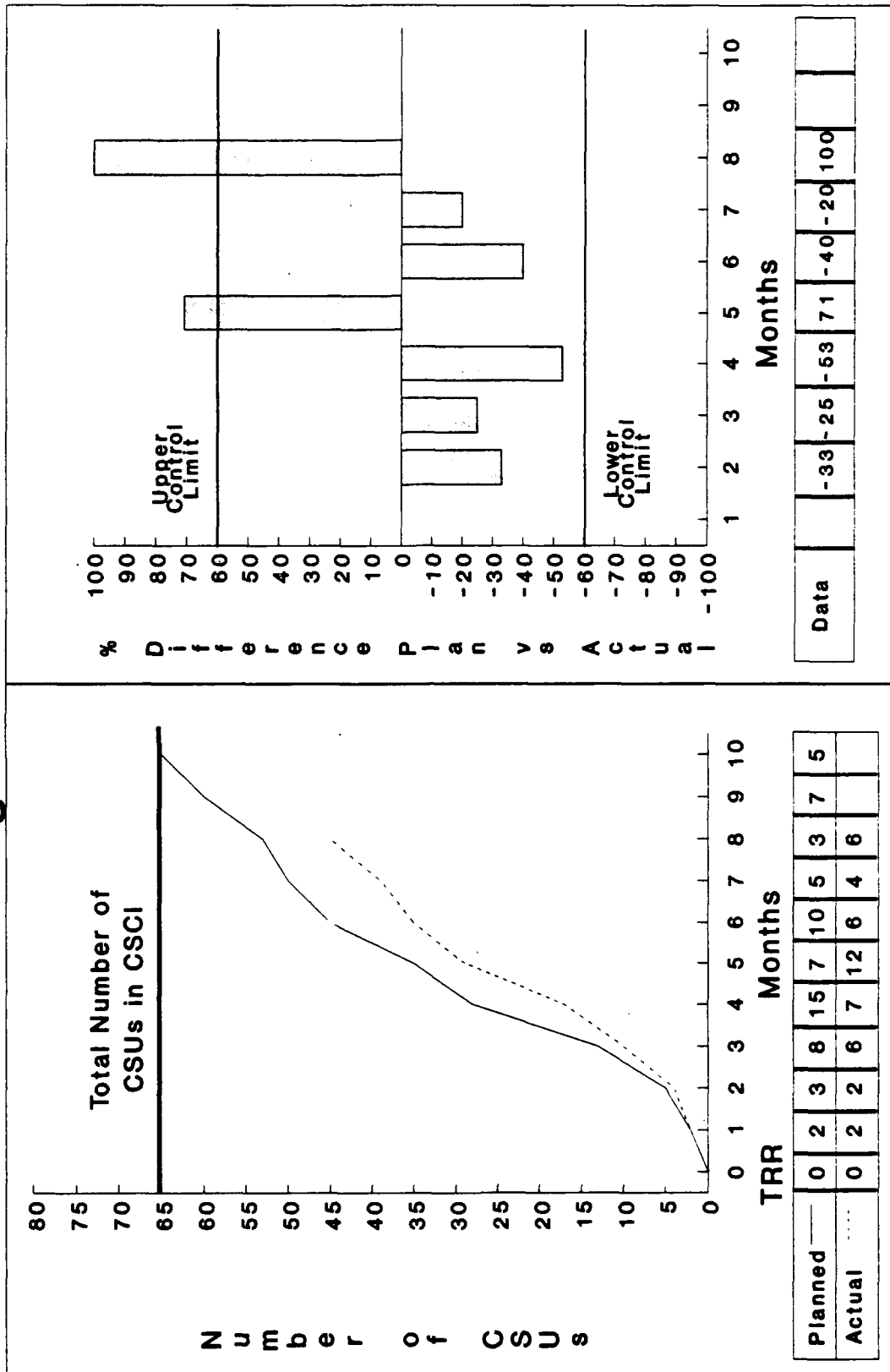
4.6.1.5.4 An Example of the Indicator. An example of the Integration Status indicator is shown in Figure 4.8.

4.6.1.5.5 Using the Indicator. The indicator provides information concerning how well the integration activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the integration effort while the bottom graph indicates how well the process is performing. The top graph also shows how many CSUs underwent retesting for each reporting period. Even though the actuals may be close to the planned, a significant portion of the actuals may be retests and not new CSUs. As indicated by this example, the top graph indicates that the integration task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.6.1.5.6 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Change time period of indicator to meet your needs.
- c. Add information concerning number of CSUs recoded or modified.

# CSCI (Name of CSCI) Integration Status



As of:

4.6.2 Cost. There are two indicators associated with the cost area: man months of effort, and software size. The man months of effort indicator provides the basis for making cost estimates. The software size indicator provides a basis for determining and checking inputs to software cost models.

4.6.2.1 Man Months of Effort The purpose of the Man Months of Effort indicator is to track the level of effort being applied to the software development effort. This information can be used to indicate problems that need corrective actions and provide a basis for improving the inputs to software cost models.

4.6.2.2 Frequency of Update. This indicator will be updated monthly from contract award until completion of the program.

4.6.2.3 Update Criteria. Data for updates will be obtained from the cost/schedule control systems or internal timekeeping of the software developer.

4.6.2.4 Indicator Inputs. The indicator is constructed from the following information:

- a. Total Number of Man Months of Effort: Data is obtained from estimates, models, proposals, etc.
- b. Planned Number of Man Months: Data is obtained from the estimated number of man months to be expended on the program each reporting period.
- c. Actual Number of Man Months: Data is obtained from actual number of man months of effort being expended for each reporting period.
- d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$



e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations (18:324).

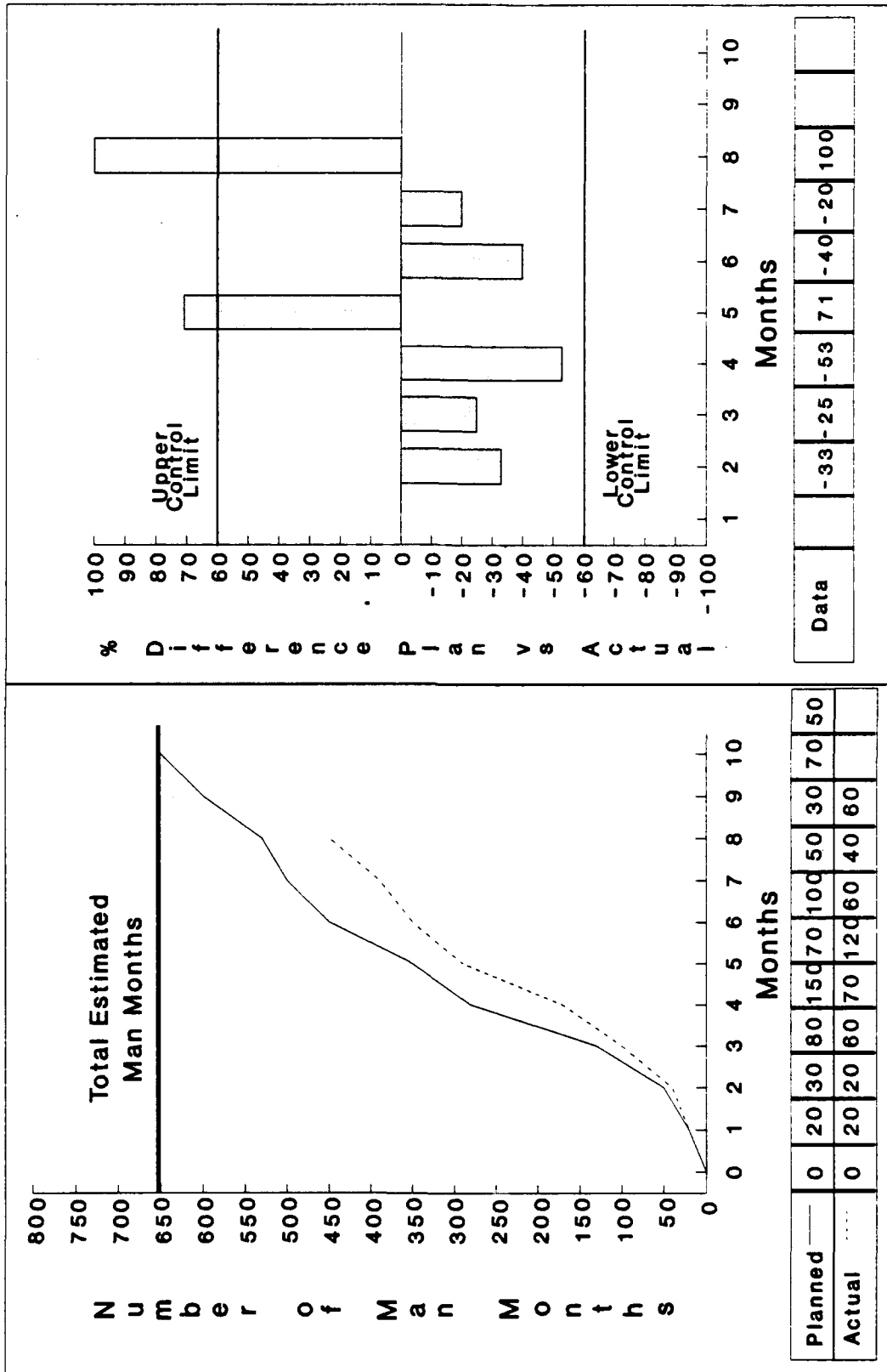
4.6.2.1.5 An Example of the Indicator. An example of the Man Months of Effort indicator is shown in Figure 4.9.

4.6.2.1.6 Using the Indicator. The indicator provides information concerning how well the actual man months match the estimated or predicted man months. In addition, it also provides information to be used to improve future estimates of levels of effort. The top graph of the indicator provides an overall status of how well the actual matches the estimate while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the integration task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.6.2.1.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Change time period of indicator to meet your needs or correspond to outputs of cost models.
- c. If Cost/Schedule Status Reports or Cost Performance Reports can provide the same data, then this indicator may be deleted. The key is to provide sufficient visibility of software within the Work Breakdown Structure to ensure the adequate information is obtained.

# CSCI (Name of CSCI) Man Months of Effort



As of:

4.6.2.2 Software Size. The purpose of the Software Size indicator is to track the size of the software being developed. This information can be used to provide a basis for improving the inputs to software cost models.

4.6.2.2.1 Frequency of Update. This indicator will be updated monthly from contract award until completion of the program.

4.6.2.2.2 Update Criteria. Data for updates will be obtained from inspections of code that has been written.

4.6.2.2.3 Indicator Inputs. The indicator is constructed from the following information:

a. Original Estimate of Total KSLOC: Data is obtained from the original estimate, model output, proposals, etc., that was derived at the beginning of the program prior to contract award.

b. Original Estimate of New KSLOC: Data is obtained from the original estimate, model output, proposals, etc., that was derived at the beginning of the program prior to contract award.

c. Original Estimate of Reusable KSLOC: Data is obtained from the original estimate, model output, proposals, etc., that was derived at the beginning of the program prior to contract award.

d. Original Estimate of Modified KSLOC: Data is obtained from the original estimate, model output, proposals, etc., that was derived at the beginning of the program prior to contract award.

e. Current Estimate of Total KSLOC: Data is obtained from the current estimate of how many KSLOC are expected at the completion of the program.

f. Planned Number of Total KSLOC to be Developed: Data is obtained from cost models, expert judgment, etc.

g. Actual Number of Total KSLOC Developed: Data is obtained from actual total number of KSLOC developed to date.

h. Actual Number of New KSLOC Developed: Data is obtained from actual number of new KSLOC developed to date.

i. Actual Number of Reused KSLOC Developed: Data is obtained from actual number of reused KSLOC to date.

j. Actual Number of Modified KSLOC Developed: Data is obtained from actual number of modified KSLOC to date.

4.6.2.2.4 An Example of the Indicator. An example of the Software Size indicator is shown in Figure 4.10.

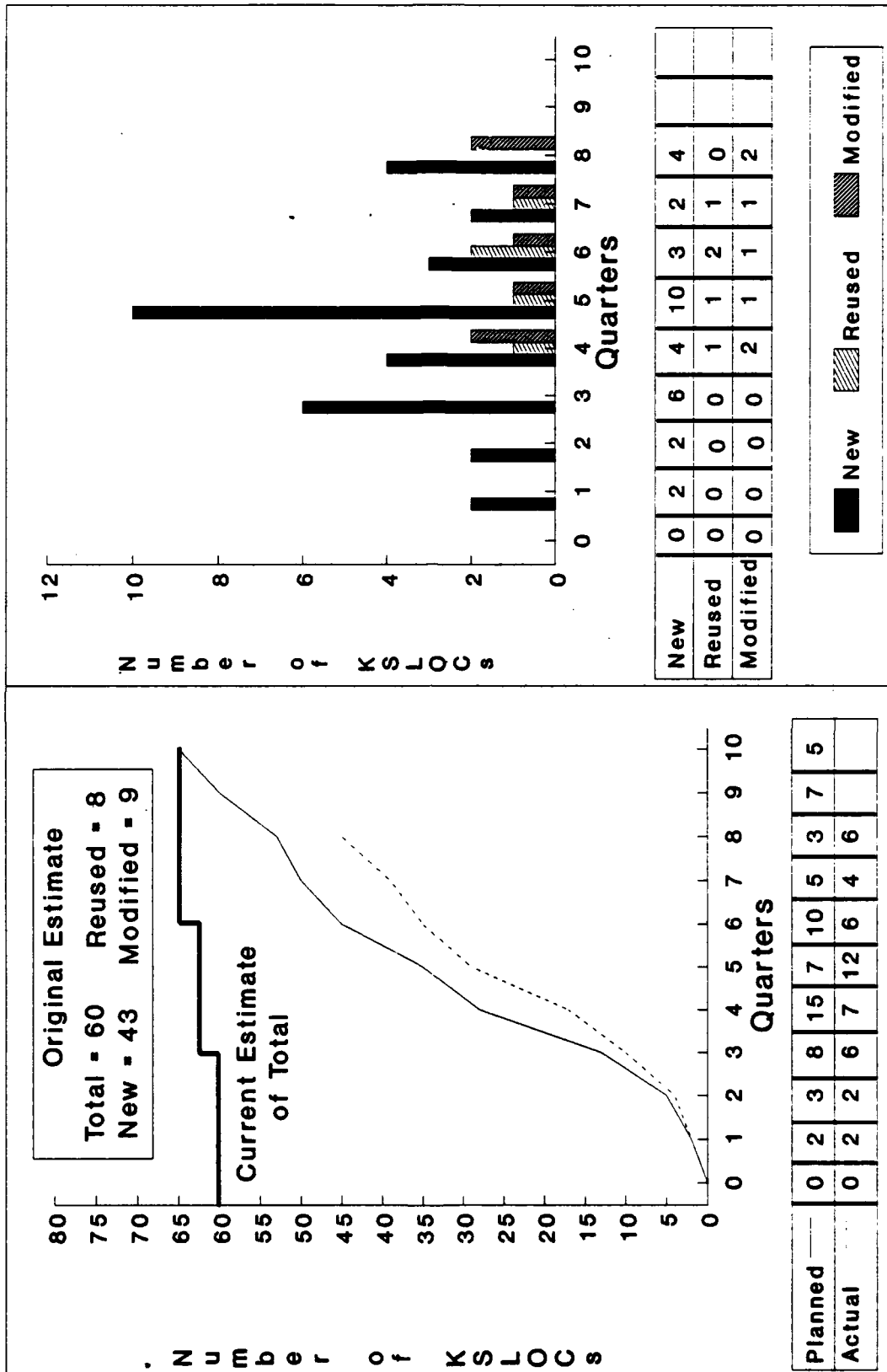
4.6.2.2.5 Using the Indicator. The indicator provides information comparing the original, current, and actual estimates of the software size. By comparing the actual to the original estimate, the accuracy of the original estimate can be determined. By comparing the actual to the current estimate gives insight into what is going on in the program and if there is need for management attention. The additional information concerning the breakout of the software between new, reused, and modified allows for comparing against the original inputs that may have been derived from cost models, and it provides data to be used to improve the inputs to and estimates from cost models in the future.

4.6.2.2.6 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Change time period of indicator to meet your needs or correspond to outputs of cost models.
- c. Eliminate the detailed breakout of the size.

# CSCI (Name of CSCI) Software Size

Figure 4.10 Example of Software Size Indicator



As of:

d. Each language, i.e. Ada, assembler, should have its own indicator to report factual information.

4.6.3 Requirements. There are two indicators associated with the requirements area: Requirements Stability and Design Stability

4.6.3.1 Requirements Stability. The purpose of the Requirements Stability indicator is to track the number of changes that are added to, deleted from, or modified within the SRS and IRS for a CSCI. Because unstable requirements can increase schedule and cost, and requirements reflect the needs of the users, this information can be used to indicate problems that need corrective actions.

4.6.3.1.1 Frequency of Update. This indicator will be updated monthly from the time the SRS and IRS are baselined until completion of the program.

4.6.3.1.2 Update Criteria. Whenever a proposed change to the SRS or IRS is formally approved, the number and type of requirements changes will provide the data for updates.

4.6.3.1.3 Indicator Inputs. The indicator is constructed from the following information:

a. Total Number of Requirements in the CSCI: Data is obtained from the requirements in the SRS and IRS plus any added requirements for the reporting period minus any deleted requirements for the reporting period.

b. Total Number of Changes: Data is obtained from the number of formally approved changes made to the SRS and IRS since they were baselined.

c. Number of Added Requirements: Data is obtained from formally approved changes to the SRS and IRS for that reporting period.

d. Number of Deleted Requirements: Data is obtained from formally approved changes to the SRS and IRS for that reporting period.

e. Number of Modified Requirements: Data is obtained from formally approved changes to the SRS and IRS for that reporting period.

4.6.3.1.4 An Example of the Indicator. An example of the Requirements Stability indicator is shown in Figure 4.11.

4.6.3.1.5 Using the Indicator. The indicator provides information concerning how stable the requirements are for the CSCI. An increasing number of changes can indicate unstable requirements that can increase schedule and cost. The breakout of the number of changes can provide increased visibility into which types of requirements are dominating the amount of changes being made.

4.6.3.1.6 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Do not breakout the types of changes.

4.6.3.2 Design Stability. The purpose of the Design Stability indicator is to track the number of changes that are added to, deleted from, or modified within the SDD and IDD for a CSCI. Because changes to the design can increase schedule and cost this information can be used to indicate problems that need corrective actions.

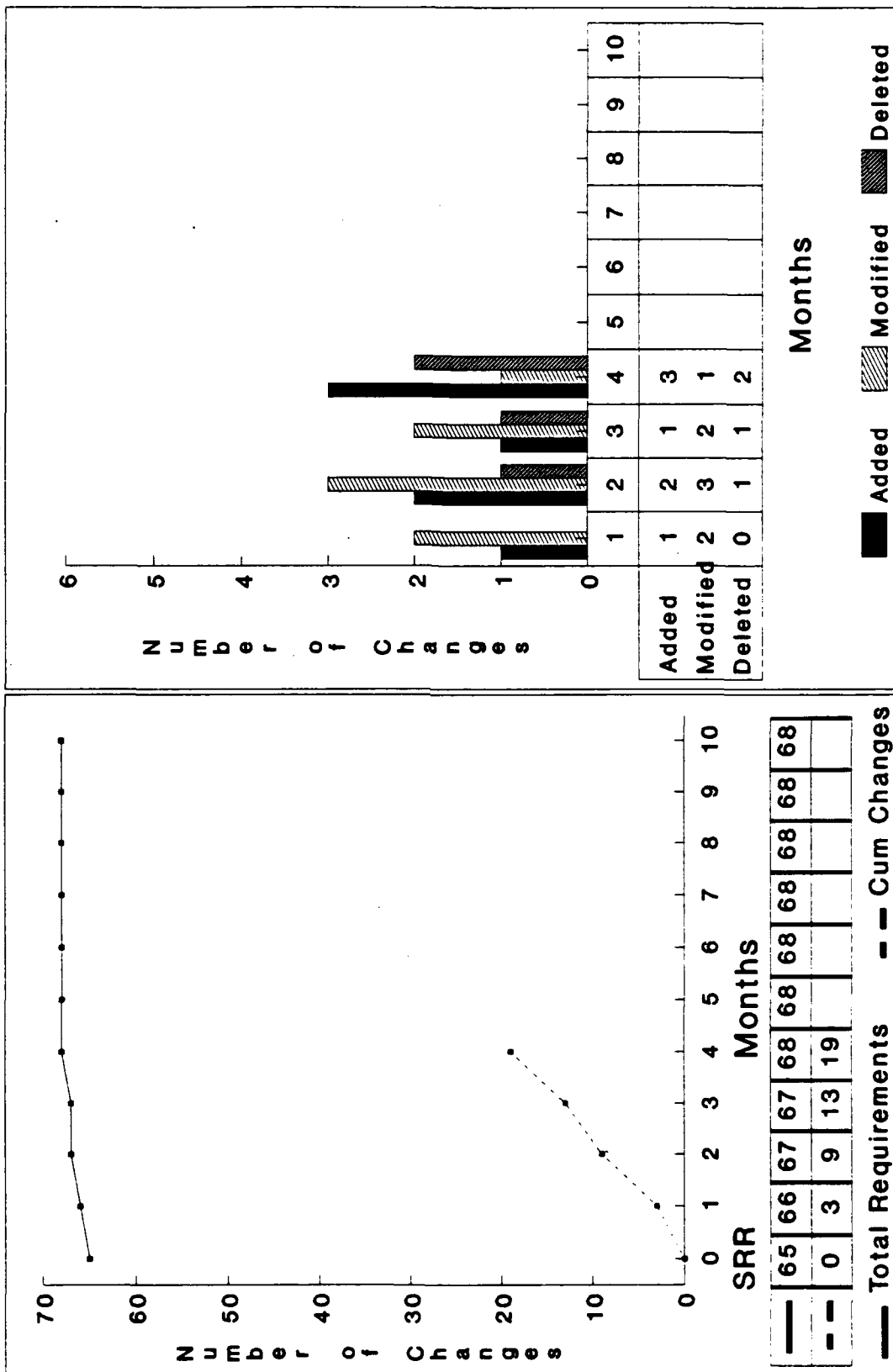
4.6.3.2.1 Frequency of Update. This indicator will be updated monthly from the time the SDD and IDD are baselined until completion of the program.

4.6.3.2.2 Update Criteria. Whenever a proposed change to the SDD or IDD is formally approved, the number and type of requirements changes will provide the data for updates.

4.6.3.2.3 Indicator Inputs. The indicator is constructed from the following information:

## CSCI (Name of CSCI)

**Figure 4.11 Example of Requirements Stability Indicator**



**As of:**



a. Total Number of Designs in the CSCI: Data is obtained from the designs in the SDD and IDD plus any added designs for the reporting period minus any deleted designs for the reporting period.

b. Total Number of Changes: Data is obtained from the number of formally approved changes made to the SDD and IDD since they were baselined.

c. Number of Added Requirements: Data is obtained from formally approved changes to the SDD and IDD for that reporting period.

d. Number of Deleted Requirements: Data is obtained from formally approved changes to the SDD and IDD for that reporting period.

e. Number of Modified Requirements: Data is obtained from formally approved changes to the SDD and IDD for that reporting period.

4.6.3.2.4 An Example of the Indicator. An example of the Design Stability indicator is shown in Figure 4.12.

4.6.3.2.5 Using the Indicator. The indicator provides information concerning how stable the designs are for the CSCI. An increasing number of changes can indicate unstable designs that can increase schedule and cost. The breakout of the number of changes can provide increased visibility into which types of requirements are dominating the amount of changes being made.

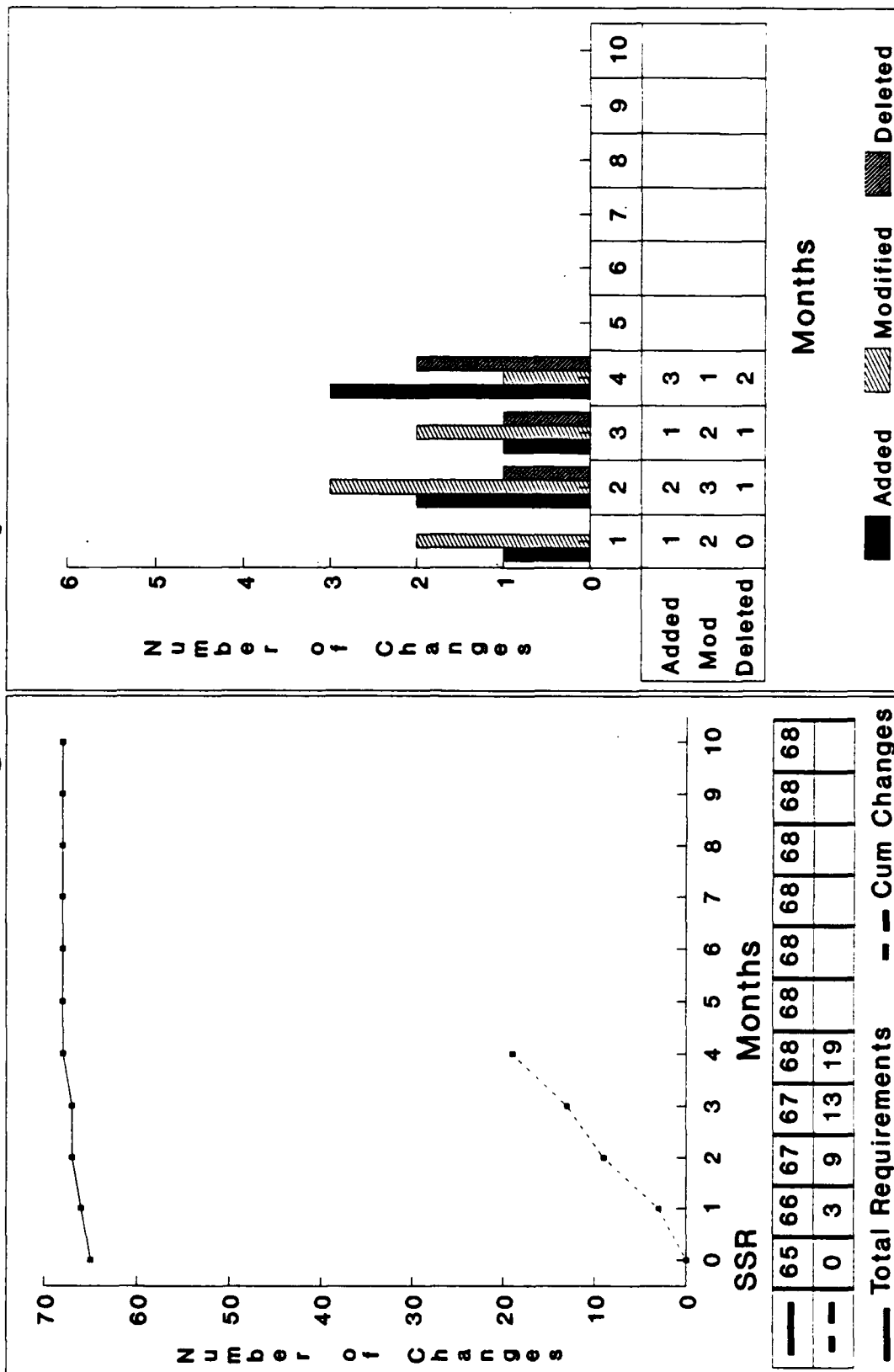
4.6.3.2.6 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Do not breakout the types of changes.

4.6.4 Software Performance. There are three indicators associated with the software performance areas: I/O Bus Throughput Capability, Processor Memory Utilization, Processor Throughput Utilization.

# CSCI (Name of CSCI) Design Stability

Figure 4.12 Example of Design Stability Indicator



As of:

4.6.4.1 I/O Bus Throughput Capability. The purpose of the I/O Bus Throughput Capability indicator is to track the estimated and actual throughput of the Bus being analyzed.

4.6.4.1.1 Frequency of Update. This indicator will be updated bi-monthly from the Software Specification Review until completion of the program.

4.6.4.1.2 Update Criteria. Estimates will be updated with actual data measured from the target hardware.

4.6.4.1.3 Indicator Inputs. The indicator is constructed from the following information:

- a. Maximum Capability of the Bus: The maximum capability of the Bus measured in KBPS and percentage.
- b. Effective Capability of the Bus: The effective capability of the Bus measured in KBPS and as a percentage of the Maximum.
- c. Management Reserve: The management reserve of the Bus throughput.
- d. Estimated Bus Throughput: The estimated throughput of the Bus.
- e. Actual Bus Throughput: The actual measured throughput of the BUS.

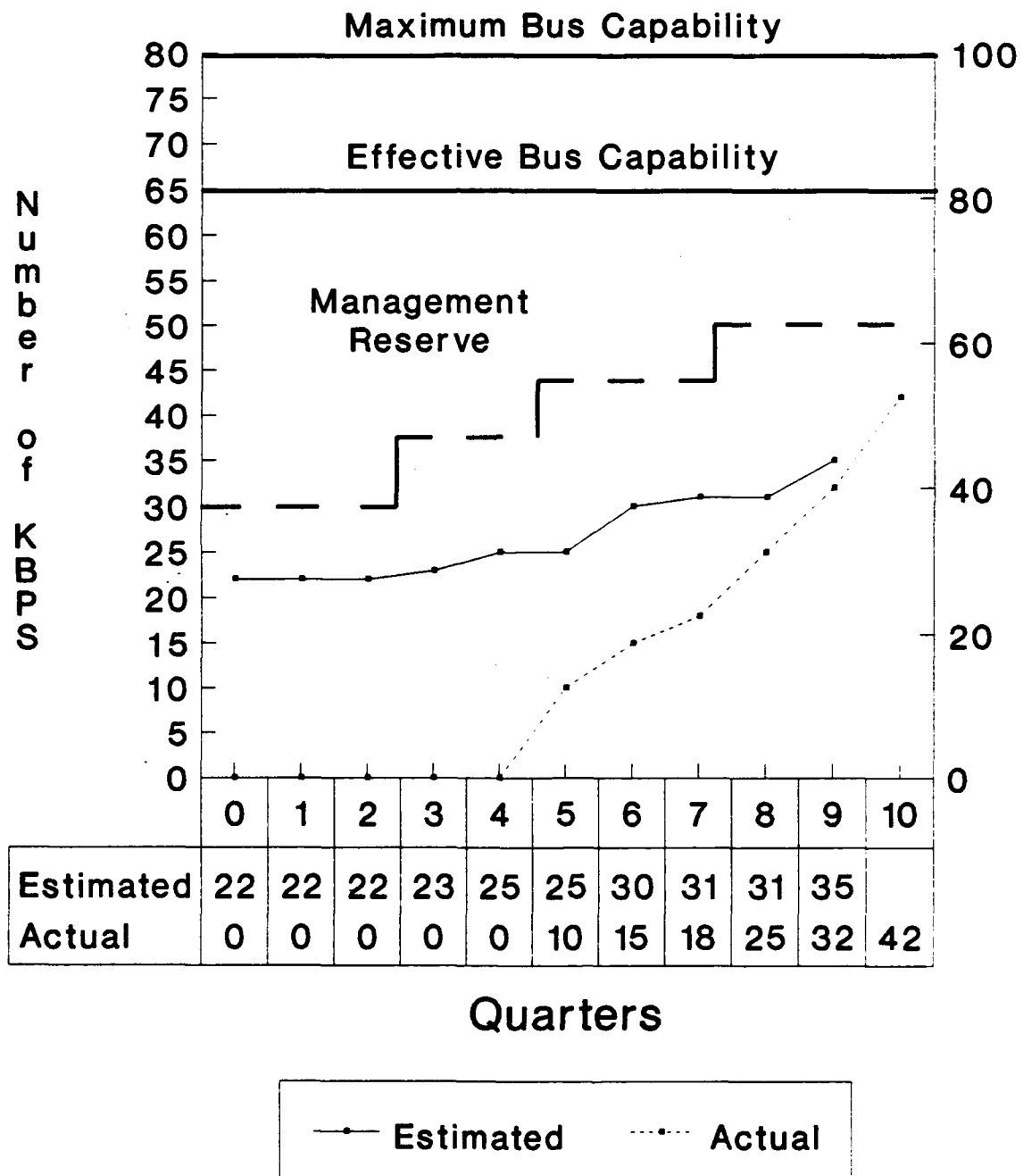
4.6.4.1.4 An Example of the Indicator. An example of the I/O Bus Throughput Capability indicator is shown in Figure 4.13.

4.6.4.1.5 Using the Indicator. The indicator provides information concerning the throughput of the Bus. It indicates when management reserve has been used up and the Bus can no longer efficiently handle the workload of the software. This allows for corrective actions to be generated to resolve the problem.

4.6.4.1.6 Tailoring Suggestions. The following suggestion may be utilized to tailor the indicator to a specific program: Change frequency of updates to meet your needs.

Figure 4.13 Example of I/O Bus Throughput Capability Indicator

## I/O Bus (Name of Bus) Throughput Capability



4.6.4.2 Processor Memory Utilization The purpose of the Processor Memory Utilization indicator is to track the estimated and actual memory usage for all the software allocated to that processor.

4.6.4.2.1 Frequency of Update. This indicator will be updated bi-monthly from the Software Specification Review until completion of the program.

4.6.4.2.2 Update Criteria. The original estimates will be updated with actual software that has been generated, modified, or reused.

4.6.4.2.3 Indicator Inputs. The indicator is constructed from the following information:

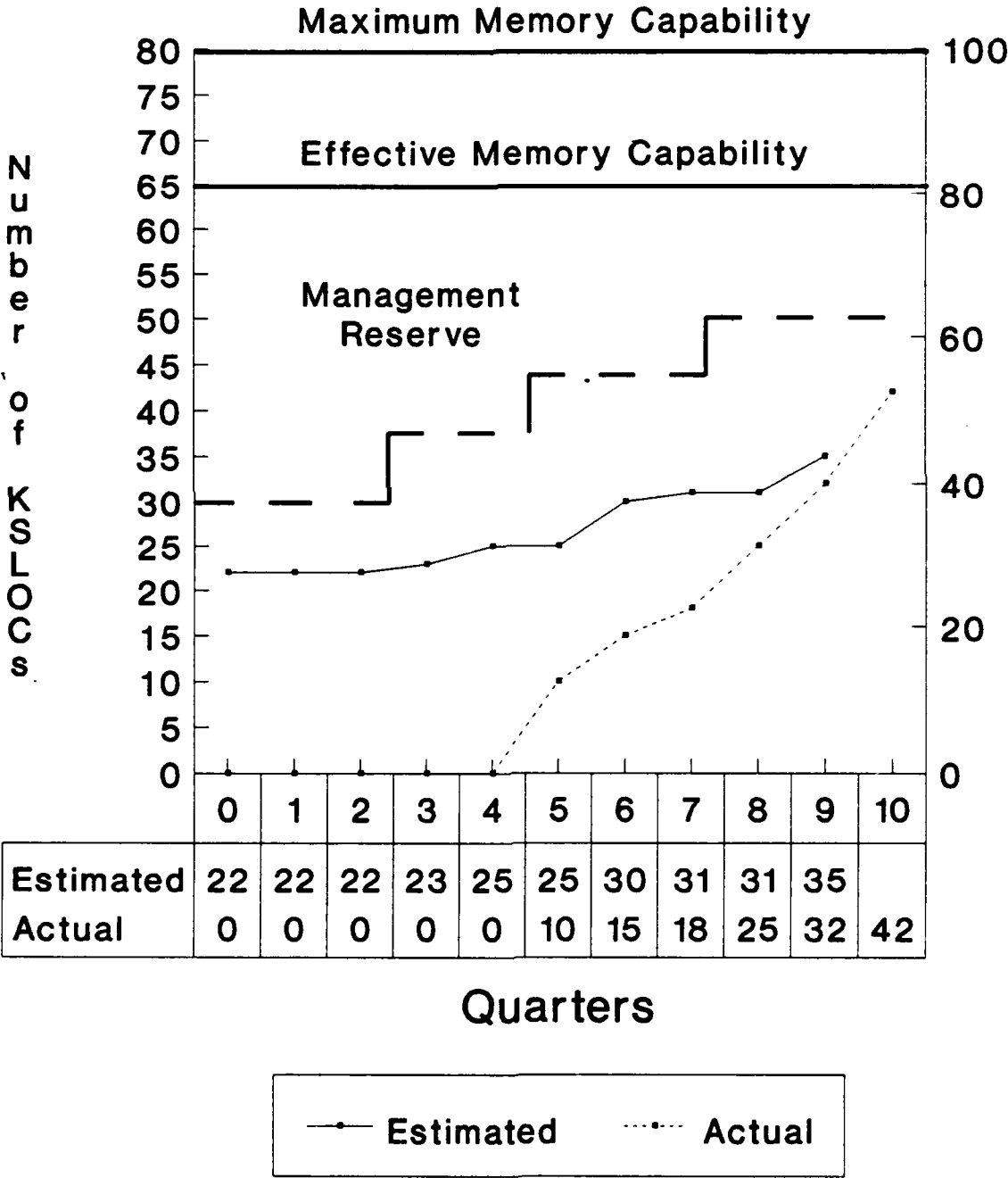
- a. Maximum Capacity of the Memory: The maximum capacity of the memory measured in KSLOC and percentage.
- b. Effective Capacity of the Bus: The effective capacity of the memory measured in KSLOC and as a percentage of the Maximum.
- c. Management Reserve: The management reserve of the memory.
- d. Estimated Memory Utilization: The estimated memory utilization of the processor.
- e. Actual Memory Utilization: The actual memory utilization of the processor.

4.6.4.2.4 An Example of the Indicator. An example of the Processor Memory Utilization indicator is shown in Figure 4.14.

4.6.4.2.5 Using the Indicator. The indicator provides information concerning the memory utilization of the Bus. It indicates when the management reserve has been used up and the processor memory can no longer handle the software. This allows for corrective actions, like rebalancing the resources, to be generated to resolve the problem.

Figure 4.14 Example of Processor Memory Utilization Indicator

# Processor (Name of Processor) Memory Utilization



4.6.4.2.6 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Breakout the memory utilization for separate CSCIs that all are allocated to the same processor.

4.6.4.3 Processor Throughput Utilization. The purpose of the Processor Throughput Utilization indicator is to track the estimated and actual execution time for all the software allocated to that processor.

4.6.4.3.1 Frequency of Update. This indicator will be updated bi-monthly from the Software Specification Review until completion of the program.

4.6.4.3.2 Update Criteria. The original estimates will be updated with actual measurements taken from the target hardware for software that has been generated, modified, or reused.

4.6.4.3.3 Indicator Inputs. The indicator is constructed from the following information:

- a. Maximum Processor Execution Capability: The maximum execution capability of the processor measured in MIPS and percentage.
- b. Effective Processor Execution Capability: The effective execution capability of the processor measured in MIPS and as a percentage of the Maximum.
- c. Management Reserve: The computational management reserve of the processor.
- d. Estimated Processor Computational Requirements: The estimated computational requirements for the processor.
- e. Actual Memory Utilization: The actual measured throughput of the processor.

4.6.4.3.4 An Example of the Indicator. An example of the Processor Throughput Utilization indicator is shown in Figure 4.15.

4.6.4.3.5 Using the Indicator. The indicator provides information concerning the throughput of the processor. It indicates when the management reserve has been used up and the processor can no longer handle the software. This allows for corrective actions, like rebalancing the resources, to be generated to resolve the problem.

4.6.4.3.6 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Breakout the memory utilization for separate CSCIs that all are allocated to the same processor.

#### 4.7 Results of Analysis of Process Problems.

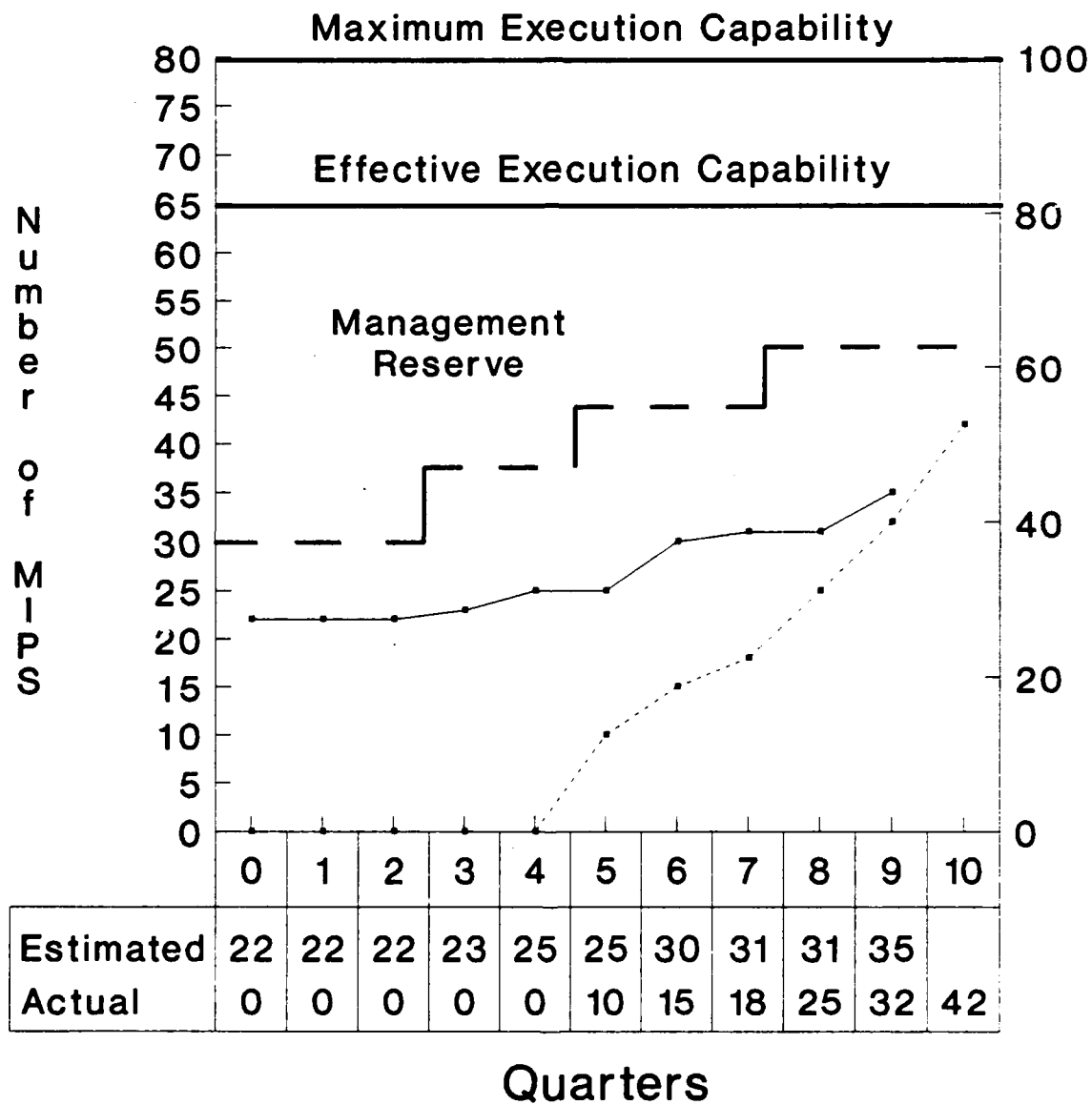
This section of the analysis compares the "expected" or ideal responses, those anticipated from an ideal software indicator environment, to those responses received in the survey. Categorization is done as to whether or not the interviewee had indicators and also the process area that is being studied (input, transformation, output, or voice of the customer). The first group analyzed is the with indicators surveys, then the without indicators surveys are analyzed with respect to both the ideal process and also with respect to the surveys with indicators.

4.7.1 Input Responses With Indicators. There were two questions that addressed the input to the software indicator environment in the with indicators survey. The first question asked for a description of the software indicator data gathering process (question 10). A large majority of the responses matched the ideal responses. There were only a few exceptions, and these responses showed that some of the software



Figure 4.15 Example of Processor Throughput Utilization Indicator

## Processor (Name of Processor) Throughput Utilization



managers and engineers were getting indicator information directly from a contractor's on-line indicator database. The second question dealing with inputs questions the types of information the interviewee gets to accomplish his or her job (question 22). The responses to this question were also overwhelmingly congruent with the responses anticipated by the ideal indicator environment.

4.7.2 Transform Responses With Indicators. There were two questions addressed to the transformation action of the software managers and engineers process. The first question asked what happens to the information provided by the software indicators (question 11). The responses divided into three basic categories: 1) information is relayed to others, 2) analyzed by the software managers and engineers, and 3) that the information resulted in some action being taken. These results are also in basic agreement with the ideal process and DODD 5000.2. The second question asked the interviewee to describe her/his current job. The responses to this question were also in agreement with the ideal software indicator environment. The question raised by the responses to these questions is if DODD 5000.2 actually gets to the root causes of the problems, or is it just stressing the identification of the symptoms? This again raises the issue of emphasis on enumerative studies versus analytic studies that was discussed in Section 4.1.1.3.

4.7.3 Output Responses With Indicators. There was a single question in the survey that dealt with the outputs of the software managers and engineers, identification of the products produced for higher level management (question 24). Almost 90 percent of the responses received were in agreement with the ideal environment. The only unusual trend noted was the lack of formal reporting identified in the responses. Only a single individual mentioned the Contractor Performance Appraisal and Reporting System as an output he generated. A trend noted in the data was a large response for "trip

reports" and "verbal updates". It is believed that this is an artifact of the examples provided by the interviewers when asking this question.

4.7.4 Voice of the Customer Responses With Indicators. There were six questions that probed the customer's (respondent's) needs from the process.

The first question asked the primary reasons why software indicators were being used. The responses again corresponded well to the ideal environment. Though there were two areas mentioned in DODD 5000.2 were not present in the responses (no mention was made of the highlighting of areas that require attention, and no one stated that the indicator data augmented conventional reports), these areas were noted in other responses to questions pertaining to the use of indicators.

The second question asked if indicators influenced the software managers and engineers ability to do their jobs (question 12A). Seventy-four percent of the respondents agreed that indicators did influence their job, and 26 percent felt that indicators did not. The primary feeling expressed as to why indicators did not influence their job was that they felt the contractor gets more use out of them than does the SPO. This reason and the others given will be considered in the handbook resulting from this research effort (Appendix G).

The third question asked about changes to the indicators being used on the program. Nine of the sixteen respondents stated that changes had occurred, with 60 percent of the responses stating the reason was to change the presentation of the indicator to better represent the desired information. Another reason given was the deletion of non-useful indicators. The next question followed on to the last and asked about any future changes envisioned to the software indicators (question 16). Only six of the sixteen respondents indicated that they anticipate changes with 78 percent of the responses stating the same reason given as in the previous question, better presentation

of information. Those that provided a reason why indicators would not be changing were mainly of the opinion that the program was in too late a stage to derive any benefit, or that the program was going well with the set of indicators currently in place.

The fifth question examining the voice of the customer asked if the information the software managers and engineers were getting differed in any way from what was desired. The responses were that 35 percent wanted more information on specific indicator areas, 27 percent wanted more information on how to use indicators, and 14 percent wanted information to make better estimates of the software development process. The first result to be seen from this data is that there exists a definite gap between what the software managers and engineers are getting right now, and what they want. The second is that this research effort addresses all three of these problem areas, the first through the development of a set of indicators that is representative of what the software managers and engineers want, the handbook that this research effort results in will contain information directly on how to apply the indicators, and better estimating being the thrust of the ASC/ENASC effort that spawned this research effort. The final question in this set was the catch-all, asking what else about indicators is felt to be important that we have not covered (question 25). The responses to this question were the same as in the previous question, in the areas that were felt to need improvement.

4.7.5 Input Responses Without Indicators. The only question to address this facet of the process in the without indicators survey asked what information the interviewee felt was needed to do his/her job. Although these programs did not use indicators, the responses showed that the interviewees received data of the same type as expected from the ideal process. In this area, therefore, there was no difference between the with- and without-indicator surveys.

4.7.6 Transform Responses Without Indicators. There were two questions in this category, the first asking how the interviewee estimated various program parameters that indicators are usually associated with: cost, schedule, quality (question 8). The responses showed that the interviewees used the inputs identified in the previous section, with the exception that reports were not identified as useful to this process. The question was worded in a manner that did not engender responses that were congruent to the questions asked in the with-indicators survey, so comparisons between them are limited. The second question in this category asked the interviewee to describe her/his job. The engineers in the survey responded with approximately the same areas that the with-indicator engineers responded, and the sole manager surveyed also had a fairly good tracking to the with-indicator responses.

4.7.7 Output Responses Without Indicators. The single question in this category asked the interviewee to identify the products produced for higher level management (question 19). The two most prevalent responses, activity status reports and briefings, tracked well with the ideal environment. Two areas that were not in the responses, but were in the ideal environment were the identification of problems and the formal reporting process. The formal reporting process responses were also absent from the with indicator surveys, and this may be from a lack of emphasis on a "formality" of the outputs in the responses that the survey team may have misinterpreted as non-existence of the property.

4.7.8 Voice of the Customer Responses Without Indicators. There were four questions in this area on the survey of without indicators programs. The first question asked why the program was not using indicators. There were two primary areas identified in the responses, cost of indicators and lack of emphasis on indicators at time of contract. These reasons were the same ones identified earlier in this analysis for the

programs that had indicators as to why they were not on contract. Mitigating factors are the same as identified then, the research effort's results in emphasizing the importance of indicators, and the minimal core set of indicators keeping costs to a minimum. The contracts in this portion of the research were also relatively older than the contracts that had indicators on contract, with an average contract date of October, 1986.

The second question in this category asked about future use of indicators (question 11). The prime reason for not using them in the future is the late stage of the lifecycle of the program that most of these efforts were in. It is also noted that the point in the contract lifecycle also has an impact on the use of indicators, and that contracts that are nearly finished will have less incentive to put them on contract.

The third question asked about the voice of the customer was if the software managers and engineers felt that they needed more information than they were presently getting (question 18). The responses to this question formed no clear consensus from the small data set.

The final question in this set was the catch-all, asking what else about indicators is felt to be important that we have not covered (question 20). The responses to this question centered around the areas of guidance in the use of indicators, and inhibitors to the use of indicators (cost, program phase, etc). These areas have already been identified as candidates for mitigation from the results of this research effort.

4.7.9 Summary of Analysis Overall, there were no discrepancies between the ideal process and the survey responses. The process generally follows the current guidance (DODD 5000.2). The point to be made is that current Air Force direction appears to provide direction that is directed toward an enumerative approach to the use of indicators versus an analytic approach. This is contrary to the literature that views indicators as a means to improve the process through analytic analysis.

#### 4.8 Comparison of Problems

A comparison between the problems identified by the question responses analysis is Section 4.1.1 and the results of the comparison against an ideal standard done in the previous section, Section 4.7. Due to the lack of identified problems from the analysis accomplished in Section 4.7, no real comparison could be made. The key problems became the set of problems identified in Section 4.1.1 which are: 1) not getting indicator data in a timely fashion, 2) indicators not meeting the needs of the respondent by either being for a different function or for use by higher-level management, 3) indicators providing data that could have been obtained through other sources, 4) believing that indicators are more beneficial to the contractor than the program office, 5) the software developer refusing to provide the data, 6) not tailoring the indicators for specific phases of the program, 7) identifying the need for knowledgeable people to interpret metrics, 8) a lack of knowledge about indicators themselves, 9) a lack of management involvement, 10) the cost of putting indicators on contract, 11) the lack of emphasis placed on indicators at the time of contract award, and 12) ambiguous definitions of indicators. Recommendations concerning these problems will be made in the next section.

#### 4.9 Solution to Problems.

Each of the key problems will be addressed separately with recommendations on possible solutions. The recommendations are based on the assumption that the software indicators are on contract.

4.9.1 Contractual Issues. The following paragraphs propose solutions to meet the types of problems pertaining to the contracting of indicators.

The solution to not getting indicators in a timely fashion may be resolved by two different methods: change reporting period or change method of delivery. The first

method can be adjusted to meet the needs of the manager or engineer but has the drawback of potentially increasing costs as shorter delivery times are mandated. The second can be accomplished several ways. One way is to have direct access to the software developer's database so that information can be obtained at any time. Another approach would be to have the software developer transmit a facsimile of the indicators which would certainly speed up the mailing process. The main point to be made is to avoid tying the delivery of indicators to a standard data item, like a monthly status report, where indicators are not the primary focus of the data item.

The solution to the software developer refusing to provide the data can be resolved by placing the task of providing indicator data on contract. Though this may not totally resolve the reluctance of the software developer to provide data, it provides a vehicle for the program office to obtain the information by making it a binding agreement between the two parties.

A solution to the cost of putting indicators on contract cannot be provided. The cost of putting indicators on contract has been minimized by the development of a core set of indicators based on those areas identified as very important to software managers, engineers, and ASC/ENASC. By following the Plan, Do, Study, and Act cycle the standard set of indicators can further be improved which should continue to make the indicators more cost-effective.

4.9.2 Selection of the Wrong Indicator. The following paragraphs propose solutions to meet the types of problems pertaining to selection of the incorrect indicator for the program.

The solution to indicators not meeting the needs of the respondents should be resolved with the standard set of indicators identified by this research effort as these were derived from the stated needs of the individuals interviewed. Though a standard



set will not satisfy all needs for every program, it certainly should provide a firm basis for supporting management of software development efforts within ASC.

The solution to the problem of indicators providing redundant information that could have been obtained from other sources is to ensure that no duplication exists between various sources of information available to managers and engineers. Through the research effort, the standard set of indicators should certainly provide information necessary to the management of the software development effort. They also provide a all the necessary information in a single product which allows for better assessment of the impacts of the interactions of the various core areas identified. If there are other sources of information available, the benefit of those sources over the advantages of having a standard set of indicators must be weighed carefully to determine which provides the most benefit for the cost.

The solution to not tailoring indicators to the different phases of the software development effort is in the guidance provided by the procedures to implement the standard set of software indicators. The procedures show a matrix of the standard indicators and program reviews and the phasing of the indicators in relation to those reviews.

The solution to the problem of ambiguous definitions of indicators may be resolved by the development of the indicator handbook. Attempts were made to define the indicators to the greatest extent possible. Recommendations are made to ASC/ENASC to work with the computer resource focal points to further add definition to the standard set of indicators and clear up any ambiguities.

4.9.3 Lack of Knowledge About Indicators. The following paragraphs propose solutions to meet the types of problems pertaining to helping software managers and engineers obtain complete information about the use of indicators.

A solution to the problem of not having knowledgeable people to interpret indicator data cannot be provided. Attempts have been made to provide details of how to use the standard indicators and interpret their data. It is a recommendation to ASC/ENASC that they have individuals to become "experts" on software indicators to provide support to managers and engineers on selecting and tailoring indicators for their programs.

The solution to the problem of a lack of knowledge of software indicators can be alleviated by the establishment of a standard indicator program within ASC. This program, combined with the handbook, should provide education to managers and engineers on indicators and their uses.

4.9.4 Other Problems With Indicators. The following paragraphs propose solutions to meet problems with the use of indicators.

The solution to the premise that indicators are only beneficial to the software developer and not managers and engineers can be resolved by noting that not all problems identified by indicators are due to the actions of the software developer and some may be directly linked to the actions of the program office. This is not to say that the software developer should not be concerned about indicators, but there should be a joint effort between the software developer and the program office to identify problems and make improvements to resolve them. This will result in a better process and therefore a better product.

The solution to the lack of management involvement needs to be addressed by top management. Top management must become involved and support data gathering efforts for them to be successful. If top management becomes involved with and supports the effort, managers throughout the organization will do the same. With the recent publication of ASDP 700-8, ASD Metrics Handbook, and the award to ASC of

the Quality Improvement Prototype Award, there appears to be an increased management focus on indicators which may mitigate this problem.

There may already be a solution in place to the problem of a lack of emphasis on indicators resulting in indicators not being placed on contract. Mitigating factors in the lack of emphasis would be the ASDP 700-8, ASD Metrics Handbook, and this research effort. There is also a marked swing in later contracts to the effort to put indicators on contract, as evidenced by the average year of contract award. For those programs with no indicators on contract, the average is March, 1984, and the average for those programs that have indicators on contract is October, 1989.

4.9.5 Implementing the Proposed Solutions. The development of a standard set of software indicators and procedures is the vehicle for implementing solutions to the types of problems associated with contractual issues and selection of the wrong indicator. The standard set is based on the needs of software managers, engineers, and ASC/ENASC and the procedures incorporate strategies for dealing with contractual issues. The establishment of a standard software indicator program is the vehicle for implementing solutions to the types of problems associated with a lack of knowledge, belief that indicators are only for the software developer, lack of management involvement, and a lack of emphasis on indicators. A standard indicator program should provide a focus for providing education and training on the use of indicators.

#### 4.10 Vision of the Improved Process.

To focus on the improved process, the first step is to look at the current environment which is then used to generate the vision of the improved process.

4.10.1 Opportunity for Improvement. The vision of the improved process is derived from the documented current environment. The previous sections have

developed a picture of what that environment is from the data that was analyzed. This environment and the opportunities for improving it are illustrated in Figure 4.16.

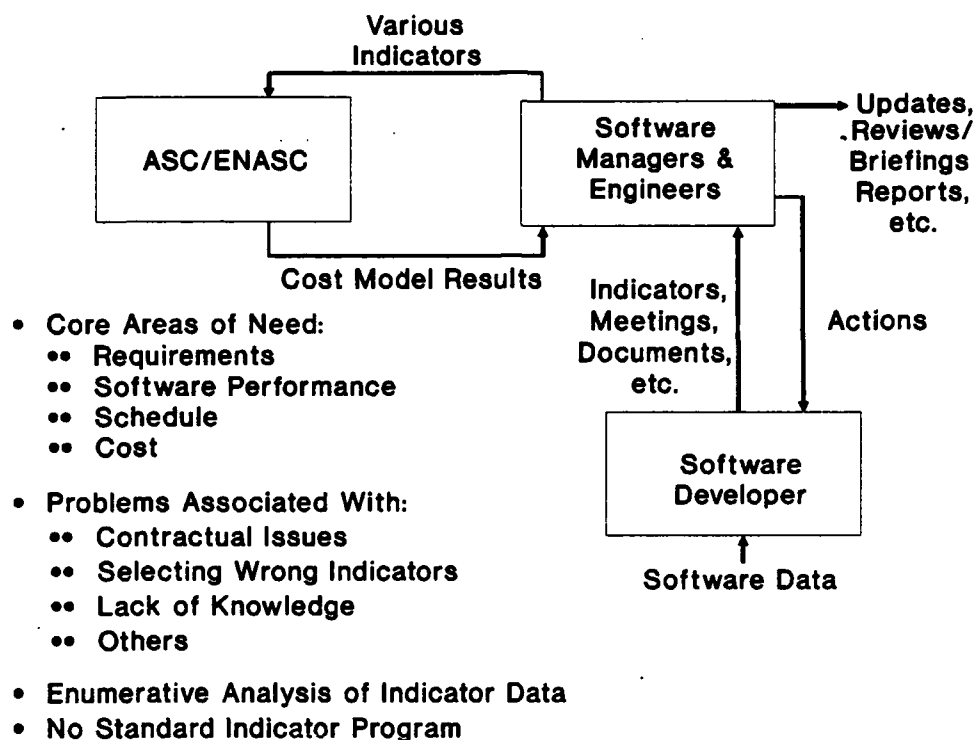


Figure 4.16 Current Environment

**4.10.2 Improving the Current Process.** The vision of the improved process centers around the establishment of a standard indicator program and a set of standard software indicators. The implementation of a standard indicator program and set of indicators should solve many of the types of problems identified in the current environment. To do this, the standard indicator program should be responsible for:

- 1) educating and training managers and engineers in the use of software indicators;
- 2) for providing technical assistance to the program offices;
- 3) providing guidance and direction to move from enumerative analysis to analytic analysis of indicator data;
- 4) maintaining the database for indicator data from all programs within ASC, and 5)

tying together the cost models, software indicators, and process models. The indicator database provides the means for developing better estimates and cost models. As the predictions and estimates get more accurate, focus can be applied to eliminating true causes of variance in the process. This leads to improved software development processes which in turn produce better software products.

Another aspect of the improved process is the encouragement of joint government/software developer product teams. Because sources of variability may be due to actions generated by both the government and software developer, a joint product team is necessary to eliminate all sources of variation.

A key to successfully accomplishing this is the relationship between the software developers and program offices, as well as the relationship between the program offices and ASC/ENASC. The software developers and program offices work together to identify problems and resolve them. The program offices and ASC/ENASC work together to improve upon the indicators to ensure that the needs of everyone involved are met. A flow diagram of the improved process is shown in Figure 4.17.

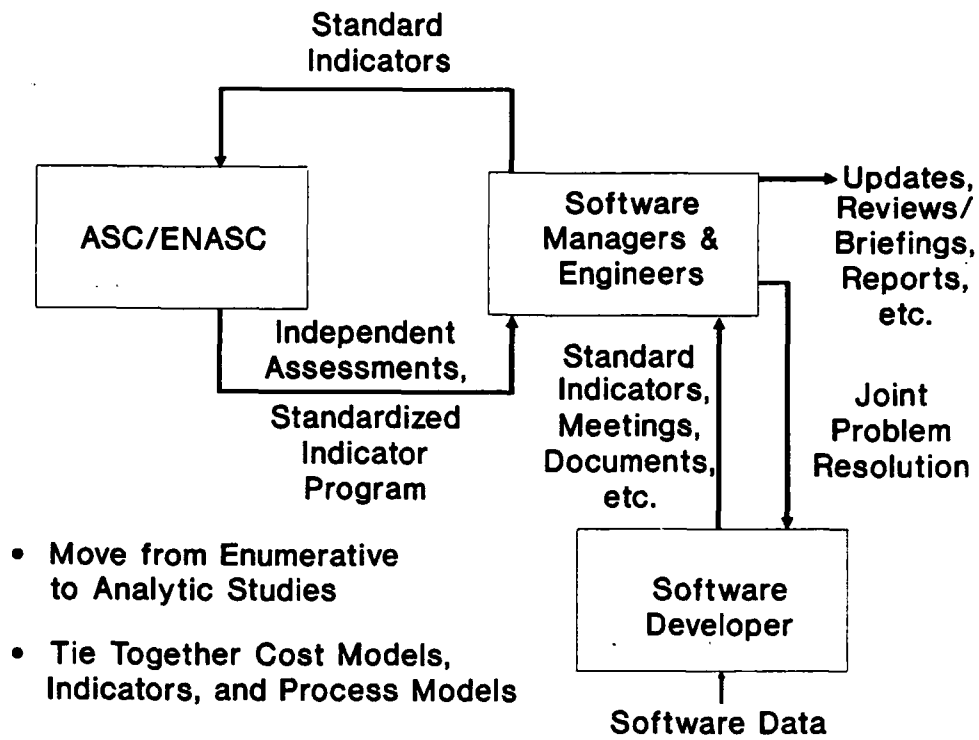


Figure 4.17 Vision of the Improved Environment

## 5.0 Conclusions and Recommendations

This chapter will present both the objectives and conclusions arrived at from the conduct of the research effort. It will also discuss the key findings from the data analysis, recommendations for the sponsors of the research, and the research team's thoughts on potential follow-on actions that would expand upon the effort, provide additional insight into the software development process and the means of improving that process.

### 5.1 Overview of the Research.

This section of Chapter 5 provides a short summary of the means by which the research was conducted.

5.1.1 Objectives of the Research. This research was initiated to satisfy the need for a standard set of software indicators at ASC. This need led the research team into a process of discovery about the needs of the organizations at ASC and the process of selecting and employing software indicators. There were three research objectives that were used to guide the research, as shown in Table 5.1.

TABLE 5.1

#### RESEARCH OBJECTIVES

1. Identify what data software managers and engineers need to improve software development efforts and what ASC/ENASC needs to populate their database. This can be stated as answering the question "What data do software managers, engineers, and ASC/ENASC need?"
2. Identify the software indicators necessary to provide the data identified by the first objective. This objective answers the question "How will the data needed by software managers, engineers, and ENASC be measured?"

TABLE 5.1

RESEARCH OBJECTIVES (CONTINUED)

3. Identify the procedures necessary to implement the software indicators identified by the second objective. This objective answers the question "What changes are required to implement the proposed software indicators?"

5.1.2 Research Methodology. The research began with a canvas of the literature on software indicators to determine a methodology for the selection of a set of indicators. Although there is extensive literature available on indicators, there are few sources that delineate the methodology for selecting indicators. The ones most frequently mentioned fell into three categories: 1) pick more than you need, and winnow out the unneeded ones; 2) assemble a panel of experts to pick them for you; or 3) development of a goal-setting team to develop questions to find out what is needed. These approaches were unsuitable for this research effort, because they were too costly (category 1), or required the assembly of a team of experts (categories 2 and 3). This led the research team to pursue an alternate route--to use an accepted general problem-solving methodology and tailor it to the job at hand. The work of W. Edwards Deming was known to the team, and Deming's well-defined problem solving technique and overall emphasis on continuous improvement appeared to be suited to the research effort. The Deming approach was operationalized for the effort, and resulted in the steps and their related tasks shown in Figure 5.1.

The research team then decomposed the high level tasks into a series of actions that would provide the information required to complete each task. For each objective, these tasks are shown in Figures 5.2 to 5.4. These three figures show the relationships of the intermediate products produced in the research to the final results of that effort.



## Research Actions

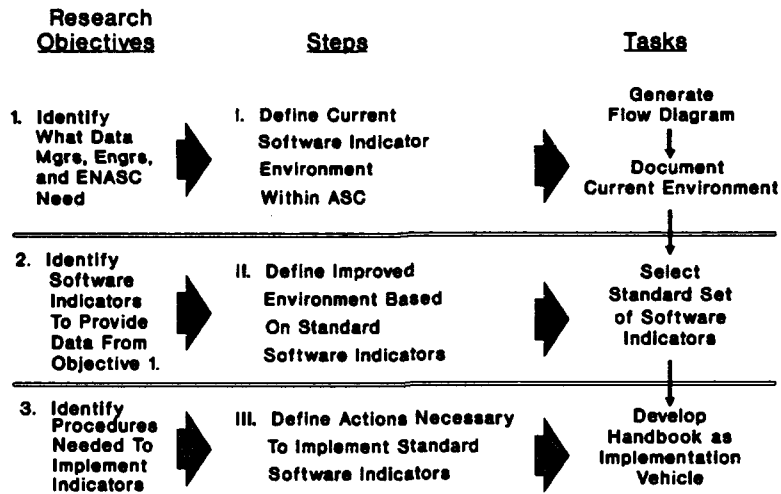


Figure 5.1 Research Objectives

The set of tasks identified in Figure 5.1 are the starting point for the creation of actions needed to conduct the research. The tasks identified in conjunction with research objective 1 are decomposed into actions as shown in Figure 5.2.

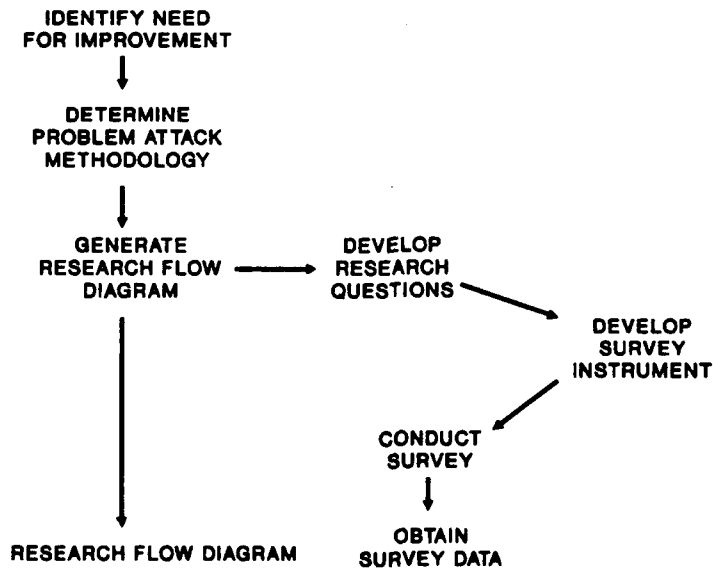


Figure 5.2 Actions for Objective 1

The actions needed to complete research objective 2 were grouped into two sets:

- 1) the actions needed to define the current environment, from the actions and data gathered in Figure 5.1, and 2) the development of the vision of the improved indicator environment. The actions to define the current environment are detailed in Figure 5.3

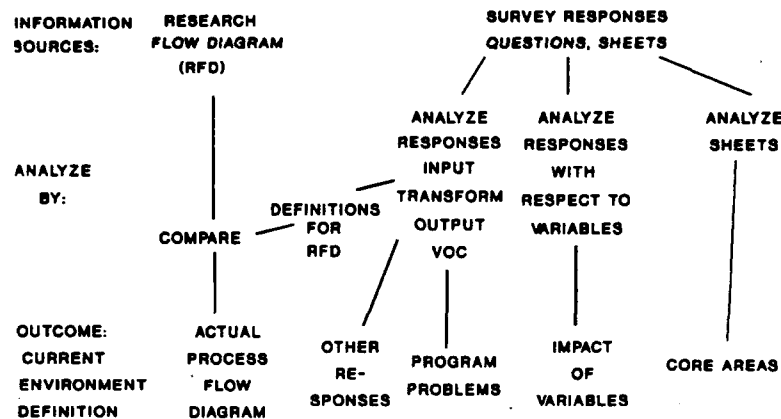


Figure 5.3 Actions to Begin Objective 2

Finally, the actions to complete research objectives 2 and 3 are shown in Figure 5.4.

Upon completion of the actions, the key element to implement the new environment (the indicator handbook, Appendix G) and the actions identified in this chapter to begin the implementing the new environment are ready for use.

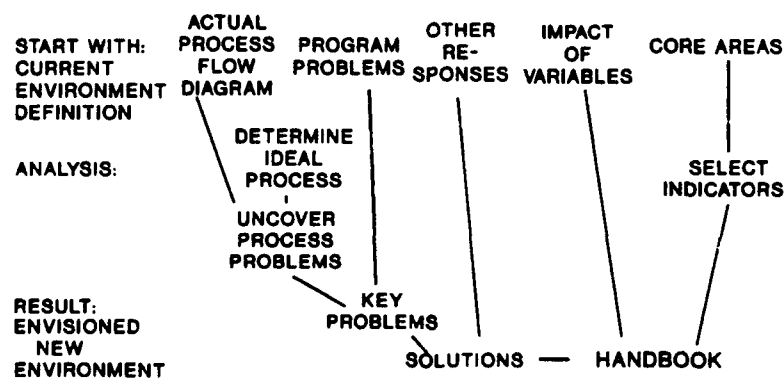


Figure 5.4 Actions to Complete Objective 2

## 5.2 Detailed Results of the Research.

The details provided here are extracted from the methodology in Chapter 3 and the data analysis from Chapter 4.

5.2.1 Results of Objective 1. The first result of the research was a detailed model of the process for the use of software indicators at ASC. This process is shown in Figure 5.5.

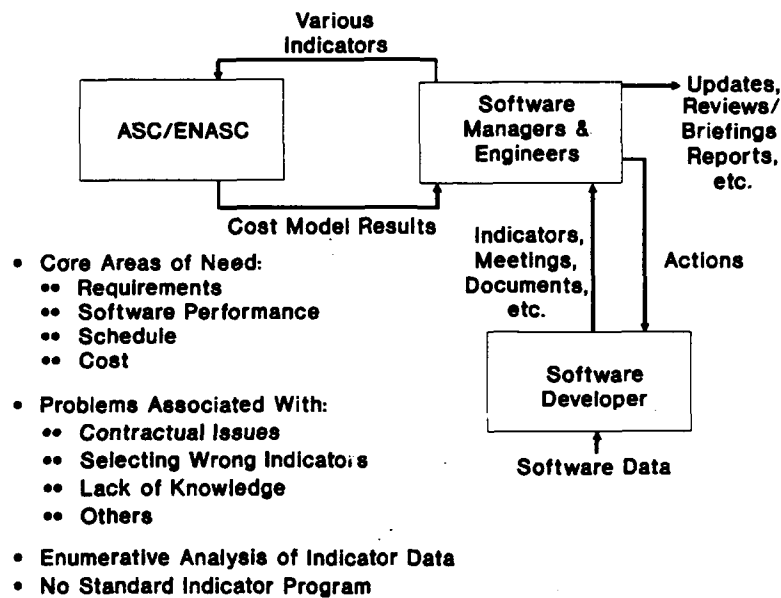


Figure 5.5 Indicator Process at ASC

In addition to the process diagram, the research also studied the needs of managers, engineers, and ASC/ENASC. Through the analysis detailed in Chapter 3, the core areas shown in Figure 5.6 were derived from the data gathered. These core areas represent the needs of the SPOs and ASC/ENASC. They also represent the fulfillment of objective 1.

5.2.2 Results of Objective 2. The second objective of the research was to identify the standard set of indicators for ASC. With the needs identified in Objective 1, the

indicators were then derived from the database of indicators developed by the research team during the review of the literature on software indicators.

SURVEY SHEETS IMPORTANT AREAS	ENGINEERS AREAS	ENGINEERS INDICATORS	ENGINEERS AREAS	MANAGERS INDICATORS	MANAGERS AREAS	ENASC AREAS
REQUIREMENTS	✓	✓	✓	✓		
SCHEDULE			✓			✓
COST						✓
SOFTWARE PERFORMANCE	✓		✓	✓		

REQUIREMENTS, SOFTWARE PERFORMANCE  
SCHEDULE, AND COST  
ARE CORE AREAS OF INTEREST

Figure 5.6 Core Areas

The indicators chosen were all from AFSCP 800-43, Software Management Indicators with the exception of the cost indicator Man Months of Effort. The form of this indicator was derived from the other cost indicator. The indicators selected are shown in Table 5.2.

TABLE 5.2  
SELECTED INDICATORS

<u>Area</u>	<u>Indicators</u>
Requirements:	CSCI Requirements Stability CSCI Design Stability
Software Performance:	I/O Bus Throughput Capability Processor Memory Utilization Processor Throughput Utilization

TABLE 5.2  
SELECTED INDICATORS (CONTINUED)

<u>Area</u>	<u>Indicators</u>
Schedule:	Requirements Allocation Status Preliminary Design Status Detailed Design Status Code and Unit Test Status Integration Status
Cost:	Man Months of Effort Software Size

Although these indicators were from the same primary source, the research team recommended details of implementation to make the presentation of the indicator more clear to the user.

5.2.3 Results of Objective 3. The changes required to implement the standard set of indicators will involve the dissemination and use of the handbook on indicators that is the result of this research. There are also recommendations to ASC/ENASC as to efforts they may want to pursue to enhance the use and utility of indicators at ASC. These recommendations are in section 5.3. It is suggested that the indicators be applied to the programs at ASC as part of an effort to create an environment of continuous improvement in the development of software.

### 5.3 Recommendations for ASC/ENASC.

The research effort reported on in this document is useless if the results are not implemented. The following is the research team's views on how ASC/ENASC should put these findings into action. As stated in Chapter 1, this research effort concentrated itself on only one part of the Deming cycle for continuous improvement, the "Plan"

indicators were then derived from the database of indicators developed by the research team during the review of the literature on software indicators.

SURVEY SHEETS IMPORTANT AREAS	ENGINEERS AREAS	ENGINEERS INDICATORS	MANAGERS AREAS	MANAGERS INDICATORS	ENASC AREAS
REQUIREMENTS	✓	✓	✓	✓	
SCHEDULE			✓		✓
COST					✓
SOFTWARE PERFORMANCE	✓		✓	✓	

REQUIREMENTS, SOFTWARE PERFORMANCE  
SCHEDULE, AND COST  
ARE CORE AREAS OF INTEREST

Figure 5.6 Core Areas

The indicators chosen were all from AFSCP 800-43, Software Management Indicators with the exception of the cost indicator Man Months of Effort. The form of this indicator was derived from the other cost indicator. The indicators selected are shown in Table 5.2.

TABLE 5.2  
SELECTED INDICATORS

<u>Area</u>	<u>Indicators</u>
Requirements:	CSCI Requirements Stability CSCI Design Stability
Software Performance:	I/O Bus Throughput Capability Processor Memory Utilization Processor Throughput Utilization

TABLE 5.2  
SELECTED INDICATORS (CONTINUED)

<u>Area</u>	<u>Indicators</u>
Schedule:	Requirements Allocation Status Preliminary Design Status Detailed Design Status Code and Unit Test Status Integration Status
Cost:	Man Months of Effort Software Size

Although these indicators were from the same primary source, the research team recommended details of implementation to make the presentation of the indicator more clear to the user.

5.2.3 Results of Objective 3. The changes required to implement the standard set of indicators will involve the dissemination and use of the handbook on indicators that is the result of this research. There are also recommendations to ASC/ENASC as to efforts they may want to pursue to enhance the use and utility of indicators at ASC. These recommendations are in section 5.3. It is suggested that the indicators be applied to the programs at ASC as part of an effort to create an environment of continuous improvement in the development of software.

### 5.3 Recommendations for ASC/ENASC.

The research effort reported on in this document is useless if the results are not implemented. The following is the research team's views on how ASC/ENASC should put these findings into action. As stated in Chapter 1, this research effort concentrated itself on only one part of the Deming cycle for continuous improvement, the "Plan"

section. The research has identified the opportunity for improvement, documented the present process of software development at ASC, built a vision of the improved process shown in the flow diagram in Figure 5.5 above, and defined the improvement effort in the form of the standard set of software indicators. The recommendation for ENASC is to begin with a validation of the set of indicators and the handbook to implement them. This validation would be done with the computer resource focal points at ASC. The main activity after this validation would be the inception of the standard indicator program, with the development of the standard procedures to use the indicators. The support of top management is considered critical to the success of any indicator program (18:304), and this is an opportune point to begin the implementation of this initiative, as the standard set of indicators echoes the directions of ASDP 700-8, ASD Metrics Handbook, and the recent Quality Improvement Plan award earned by ASC. The metrics handbook focuses on constant improvement (1:5), which is the same direction this research effort advocates to ensure that the data gathering process is as complete as possible. This philosophy in ASDP 700-8 can be the basis for obtaining the resources needed to carry out the increased interaction between ENASC and the ASC software managers and engineers.

Education must be the backbone of any attempt to implement this standard set of indicators and the standard procedures to use. Several of the surveys identified the lack of education or understanding as a major problem with or obstacle to the successful implementation of indicators.

ASC/ENASC should continue with the next step in the Deming cycle, the implementation of the suggested plan on a small scale in an actual software development environment. This would involve implementation of the program outlined in the handbook on an actual software development to provide an opportunity to implement



and monitor the improvement of the process hypothesized in the research effort. The quantification of the benefits of indicators will provide additional proof of the ability of a standard set of indicators to save money. At the same time, the indicator information presently available should be used to help calibrate cost models for indicators, a major use for the services of ASC/ENASC at the present time. This effort was also suggested during the interviews of ASC software managers and engineers who had been helped by ENASC (Appendix D.4). Cost estimation and the use of cost models was one of the prime benefits identified by these interviewees, and many of the models used are able to be calibrated to better match the environment in which they are being used. Another avenue for ENASC to pursue include doing trend analyses of the data being gathered.

The main focus of these recommendations is for ENASC to increase their interaction with the SPOs.

#### 5.4 Recommendations for Future Research Efforts.

As identified in chapter 1, this research effort was limited in scope by time and the need to begin with a characterization of the process under examination. Recommendations for future research efforts are predicated on using the effort outlined in this document as a starting point to enable future efforts to delve more deeply into the software development process and the use of software indicators at ASC. Efforts that could be accomplished in this area include:

**Prove Hypotheses.** There are several hypotheses in the research effort, and these could be researched in more detail and the results used to define the process flow diagram to a greater degree.

**Impacts of Variables.** As seen in chapter 4, the research team hypothesized the impact of several variables on the software development process. Given the broad scope

of the research, small sample sizes in many instances, and time limitations, none of these variables could be statistically proven to impact the process. Another research effort might concentrate on candidate variables and perform a deeper analysis of the situation.

**Added Value of Indicators.** An analysis of the costs involved in collecting indicators could be done in conjunction with the effort suggested for ENASC above, or as a separate effort, to provide quantitative evidence of the benefits of indicators. This could be done as a case study of a program with a history of indicator collection and cost tracking.

**Work With ENASC.** Other projects could also be undertaken with ENASC, depending upon their needs. A study could be conducted with the ENASC implementation of the indicator handbook to determine the veracity of the improved process postulated in this research.

The process that this research has examined requires some refinements to be able to better do the job of producing software. Some of the avenues to be explored to accomplish this have been identified, but the main driving requirement for improvement is the gathering of data. As Watts Humphrey says "Since the debate on data definitions and formats can be endless, the only way to find out how to gather and analyze software data is to gather and analyze software data." (18:333)

With the focus of this research effort, ASC/ENASC has the tools needed to start that process.

### Appendix A: Indicator Database

This database of indicators was derived from various publications. It includes the source(s) of the indicator, whether the indicator is considered a process or product indicator, the primitives used to compute the indicator, and a brief synopsis of how the indicator is computed. Some entries do not have all of the fields filled.

Indicator Name

Actual Problem Reports vs Estimated Defects

Indicator Class (Process or Product) Process

First Source AFSCP 800-49 (draft)

Second Source

additional sources

Primitives of Indicator

Problem Reports, Estimates of Defects

Prime Use of Indicator is: Quality

Additional Uses for Indicator

Cost, Management Control, Faults

How Indicator is Computed

---

Indicator Name

Bang

Indicator Class (Process or Product) Product

First Source DeMarco, controlling the software Process

Second Source

additional sources

Primitives of Indicator

fp fpm de dei deo der ob re st tr tci rei

Prime Use of Indicator is: Schedule

Additional Uses for Indicator  
Cost, Complexity

How Indicator is Computed  
primitives are computed for the project are weighted and summed.  
Details on primitives:

FP = count of functional primitives

FPM = count of "modified manual function primitives", amount of change that goes on in the real world to accommodate the software.

DE = count of data elements in system.

DEI = count of input data elements

DEO = count of output data elements

DER = count of data elements stored by system.

OB = count of objects in system.

RE = count of relationships in system data model.

ST = count of states in state transition model.

TCi = count of data tokens at boundary of i-th functional primitive, evaluated for each primitive, tokens being items not subdivided in the primitive.

REi = count of relationships involving the i-th object of the data model.

---

Indicator Name  
COCOMO

Indicator Class (Process or Product) Process

First Source SEI-CM-12-1.1

Second Source  
additional sources

Primitives of Indicator  
KLOC, cost driver multipliers, constants for development type

Prime Use of Indicator is: Cost

Additional Uses for Indicator

How Indicator is Computed

The model is used to develop cost estimates based on lines of code estimates.

Primitives:

a, b = constants (from a chart for mode and model type)

S = "value" of the source lines of code

m = a multiplier based on 15 cost driver estimates (personnel experience, tools, etc)

the estimate E is equal to  $a*(S**b)*m$ .

---

Indicator Name  
COPMO

Indicator Class (Process or Product) Process

First Source SEI-CM-12-1.1

Second Source  
additional sources

Primitives of Indicator  
constants, KLOC, Personnel level

Prime Use of Indicator is: Cost

Additional Uses for Indicator

How Indicator is Computed

Model is designed for use on large projects.

Primitives are:

a,b,c,d = constants taken from empirical data regression analysis. b and c are based on complexity class of software.

S = program size estimate, KLOC

P = estimated average personnel level over life of project

Cost E is equal to

$a + b*S + c(P**d)$

Indicator Name \_\_\_\_\_  
Cause and Effect Graphing

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
List of causes, List of effects, Aex, Atot

Prime Use of Indicator is: Testing

Additional Uses for Indicator  
Completeness, Rqmts Traceability, Reliability, Maintainability

How Indicator is Computed

Aex = number of ambiguities in a program remaining to be eliminated

Atot = number of ambiguities in program identified

Graph is combinatorial logic network.

Steps to construct:

Identify system requirements and divide into entities.

analyze requirements to identify causes and effects

assign causes and effects unique identity numbers.

represent causes and effects by nodes labeled with identifiers.

interconnect nodes from semantic analysis and Boolean logic to determine states of causes and what conditions effects will be present.

Annotate graph with constraints that are impossible due to semantic or environment constraints.

Identify ambiguities as causes without effects, effects without causes, or combinations of causes and effects that are inconsistent with spec or impossible to achieve

Measure cause/effect percentage =

$$CE(\%) = 100(1 - A_{ex}/A_{tot})$$

Test cases can be derived from graph conversion into decision table, causes X effects.

---

Indicator Name

Combined HW and SW system Op Availability

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source

additional sources

Primitives of Indicator

Prime Use of Indicator is: Reliability

Additional Uses for Indicator

Quality

How Indicator is Computed

Indicator is composed of HW and SW observed failure rates, estimated SW faults remaining and probabilities of correcting a fault when observed. Calculations are based on a Markov process to predict availability as a function of operating time.

---

Indicator Name  
Complexity, or Cyclomatic complexity

Indicator Class (Process or Product) Product

First Source SW Qual Indic, Sci Sys Inc

Second Source IEEE 982  
additional sources SEI-CM-12-1.1 ESD TR 88-01

Primitives of Indicator  
Information Flow, Mc'Cabes, or Halsetad is recommended  
(Henry/Kafura)

Prime Use of Indicator is: SW Characteristic

Additional Uses for Indicator  
Complexity, Cost, Maintainability, Coverage

How Indicator is Computed

Metric is recommended to be applied to top level CSCs, and tracking the number with high complexity. Henry/Kafura is recommended to determine what is "high" complexity.

For detailed design, Sci Systems recommends using McCabe's, as it is easier than Information flow. (but you could do IF if you wanted to)

For code and test, they recommended Halsetads and McCabes to measure complexity

IEEE, SEI: compute with  
N = number of nodes  
E = number of edges  
SN = number of splitting nodes  
RG = number of regions



Use strongly connected graph of module ( connect exit and entry node of module to "strongly" connect)

complexity  $C = E - N + 1$   
or= RG  
or= SN + 1

SEI speaks to extensions of C, with upper and lower bounds given for the value, and also of adding data declarations to the C value, calling the new indicator flow complexity.

In IEEE 982 #25 and SEI, they use the standard definition of Information flow:

lfi = local flows into a procedure  
datain = number of data structures that procedure retrieves data from  
lfo = local flows out of procedure  
dataout = number of data structures procedure updates  
length = number of non-comment source statements in a module

Local flows are defined by calling hierarchy

fanin = lfi + datain  
fanout = lfo + dataout

Information flow complexity is

$IFC = (fanin * fanout)^{**2}$

weighted IFC = length\*(fanin\*fanout)\*\*2

ESD TR: Recommends using McCabe's, with the rule that 90% of the problems happen in the 10% of the most complex modules.  
Tracking is of the top 10% of CSUs and CSCs, and the complexity of the most complex CSCI.

Indicator Name \_\_\_\_\_  
Computer Resource Utilization

Indicator Class (Process or Product) Product Process

First Source AFCSP 800-43 (86)

Second Source ESD TR 88-01  
additional sources

Primitives of Indicator  
Pct Use Memory, CPU I/O Change Magnitude, Resource limits,  
planned use of resource

Prime Use of Indicator is: SW Performance

Additional Uses for Indicator  
Cost, Complexity, Maintainability

How Indicator is Computed

AFSCP 800-43: Metrics are computed by comparing the maximum possible deliverable resources for the particular element (CPU, memory, I/O) against the minimum deliverable resource that the contractor maintains is necessary to implement specified system requirements. Actual utilization is amount of deliverable being used.

Frequency of data collection is dependent on criticality of metric.

DID references are in source.

ESD TR: Primitives are planned spare for a given resource, estimated/actual percentage of CPU, memory, and I/O channel use. Note is made that when use exceeds 70%, performance deteriorates for real-time applications.

Indicator Name \_\_\_\_\_  
Cost Benefit of Inspection

Indicator Class (Process or Product) Process

First Source AFSCP 800-49 (draft)

Second Source  
additional sources

Primitives of Indicator  
costs to close problem reports

Prime Use of Indicator is: Cost

Additional Uses for Indicator  
Schedule, Faults

How Indicator is Computed

Indicator Name \_\_\_\_\_  
Cost/Schedule Deviations

Indicator Class (Process or Product) Process

First Source AFSCP 800-43 (86)

Second Source  
additional sources

Primitives of Indicator  
CPR, C/SSR, BCWP, ACWP, LRE, BAC

Prime Use of Indicator is: Schedule

Additional Uses for Indicator  
Cost, Management Control

How Indicator is Computed

Primitives are:  
CPR = Cost Performance Report  
C/SSR = Cost/Schedule Status Report  
BCWS = Budgeted Cost of Work Schedule  
BCWP = Budgeted Cost of Work Performed  
ACWP = Actual Cost of Work Performed  
LRE = Latest Revised Estimate  
BAC = Budget at Completion

Computed indicators are:

Cost Variance = BCWP - ACWP

Schedule Variance = BCWP - BCWS

Cost Performance Index (CPI) = BCWP/ACWP

Schedule Performance Index (SPI) = BCWP/BCWS

Estimate at Completion (EAC) = ACWP + 
$$\frac{(BAC - BCWP)}{(.2*SPI + .8*CPI)}$$

Variance at Completion (VAC) = EAC - BAC

Percent Completed = BCWP\*CUM/BAC

Data collection is recommended monthly.

DODI 7000.2 is used to analyze data. Additionally, AFSCP 173-5 implements 7000.2, providing details on cost tracking methods.

Unique rules for applying data are to be developed by each product division.

DID references are in source

Indicator Name \_\_\_\_\_  
Cumulative Failure Profile

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
F

Prime Use of Indicator is: Reliability

Additional Uses for Indicator  
Faults, Quality

How Indicator is Computed

$F_i$  = failures of a given severity level for a given time interval

Graphical interpretation of failures, as plotted against a suitable time base.

---

Indicator Name  
Defect Age Profile

Indicator Class (Process or Product) Process

First Source SW Qual Indic, Sci Systems Inc

Second Source  
additional sources

Primitives of Indicator  
Data gathered from problem reports

Prime Use of Indicator is: SW Process

Additional Uses for Indicator  
Schedule, Management Control

How Indicator is Computed

---

Indicator Name  
Defect Density

Indicator Class (Process or Product) Process

First Source SW Qual Indic, Sci Systems Inc

Second Source IEEE 982  
additional sources AFSCP 800-14

Primitives of Indicator  
SWQI: Data from Problem reports. IEEE: D (defects) KSLOD (design)  
KSLOC, I (insp)

Prime Use of Indicator is: Quality

Additional Uses for Indicator  
Faults, Reliability, Cost

How Indicator is Computed

From IEEE 982:

$D_i$  are unique defects found in  $i$ -th design or code inspection

$I$  is total number of inspections

KSLOD are design phase source lines of design statements

KSLOC are implementation phase source lines executable code and data declaration code.

DD: (sum over  $I$ ) of  $D_i$

-----  
KSLOD

AFSCP 800-14: Graphs are of cumulative defects encountered divided by the total number of units in the CSCI and the cumulative defects corrected divided by the total number of units per CSCI. Hints are given to interpret values.

Indicator Name \_\_\_\_\_  
Defect Detection Efficiency

Indicator Class (Process or Product) Process

First Source SW Qual Indic, Sci Systems Inc

Second Source  
additional sources

Primitives of Indicator

Percentage detectable SW defects found in stage vice stage after where it should

Prime Use of Indicator is: SW Process

Additional Uses for Indicator  
Reliability, Quality, Coverage, Faults

## How Indicator is Computed

I guess it is measured against requirements defects found after requirements analysis phase for the prelim design, and in each succeeding stage is the number found in the stage against the number found in later stages that could have been found in the previous stage.

Indicator Name \_\_\_\_\_  
Defect Distribution

Indicator Class (Process or Product) Product Process

First Source AFSCP 800-49 (draft)

Second Source  
additional sources

Primitives of Indicator  
various categorizations of defects

Prime Use of Indicator is: Quality

Additional Uses for Indicator  
Schedule, Cost, Management Control

## How Indicator is Computed

Indicator Name \_\_\_\_\_  
Defect Indices

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source

additional sources

Primitives of Indicator  
Di Si Mi PS Ti W1 W2 W3

Prime Use of Indicator is: Quality

Additional Uses for Indicator  
Faults, Consistency, Management Control

How Indicator is Computed

Di = total defects detected in i-th phase  
Si = number of "serious" defects found  
Mi = number "medium" defects found  
Ti = number "trivial" defects found  
W1 = weight factor for serious defects (default 10)  
W2 = weight factor for medium defects (default 3)  
W3 = weight factor for trivial defects (default 1)  
PS = size of product for i-th phase  
PI = Phase index  
DI = defect index

$$PI_i = W1(Si/Di) + W2(Mi/Di) + W3(Ti/Di)$$
$$DI_i = \text{sum over } i \text{ of } (i \cdot PI_i) / PS$$

Designed so that later phases carry more weight than earlier phases.

Indicator Name \_\_\_\_\_  
Defect removal rate

Indicator Class (Process or Product) Process

First Source SW Qual Indic, Sci Systems Inc

Second Source  
additional sources

Primitives of Indicator  
Defects removed as counted from problem reports



Prime Use of Indicator is: SW Process

Additional Uses for Indicator  
Schedule, Management Control, Faults

How Indicator is Computed

---

Indicator Name  
Design / Unit Completion

Indicator Class (Process or Product) Process

First Source SW Qual. Indic, Sci Sys Inc

Second Source ESD TR 88-01  
additional sources AFSCP800-43(90)

Primitives of Indicator  
Number of designs completed for a single stage  
SSS, SSDD, SDR, CSCs, SRS, PDR

Prime Use of Indicator is: Schedule

Additional Uses for Indicator  
Rqmts Traceability

How Indicator is Computed

SW Qual Indic: Indicator referenced for several stages,  
completion meaning having passed the review for that stage

In prelim design phase, Progress is measured by number of top  
level  
designs completed. Called a quality indicator in reference

In detailed design stage, progress is number of detailed designs  
completed.

In code and test phase, it is number of units that have been coded and passed unit test.

ESD TR:

DESIGN PROGRESS:

Primitives:

SSS = system/segment specification  
review

SDR = system design

SRR = software requirements review  
review

PDR = preliminary design

CSC = computer software components  
design document

SSDD = system/segment

SDD = software design documents

Metric attempts to track orderly progression of high level design documents into actual SRSs and SDDs. By tracking this schedule, visibility is provided into the contractors ability to keep the SSR and PDR on schedule.

Inputs are number of SSDD requirements to be documented in SRS each month, and number of SRS requirements to be documented as CSCs and SDDs each month.

CSU DEVELOPMENT:

CSCI = computer software configuration item; CSU = computer software unit

Metric compares schedule of CSCIs and CSUs to be developed, design, tested, and integrated against what work is in fact done in the specified period. Note that very large CSCIs should be monitored separately.

AFSCP 800-43(90)

Indicator name is REQUIREMENTS AND DESIGN PROGRESS

Application is same as ESD TR; tracking planned vs. actual of: SSDD reqmts into SRS, SRS into the SDD, SSDD interfaces into interface reqmts specification(IRS), and from IRS into interface design documents (IDD).

---

Indicator Name  
Design Progress

Indicator Class (Process or Product)

First Source See Design/Unit Completion ind

Second Source  
additional sources

Primitives of Indicator

Prime Use of Indicator is:

Additional Uses for Indicator

How Indicator is Computed

Indicator Name \_\_\_\_\_  
Design Structure

Indicator Class (Process or Product) Process Product

First Source IEEE 982

Second Source AFSCP 800-14  
additional sources

Primitives of Indicator  
P1 P2 P3 P4 P5 P6 P7 D1 D2 D3 D4 D5 D6

Prime Use of Indicator is: SW Characteristic

Additional Uses for Indicator  
Quality, Maintainability, Reliability, Complexity

How Indicator is Computed

IEEE Primitives:

P1 = Total number of modules in program  
P2 = number of modules dependent on input or output  
P3 = number of modules dependent on prior processing (state)  
P4 = number of database elements

P5 = number of non-unique database elements  
P6 = number of database segments (partitions of the state)  
P7 = number of modules that are not single entry/single exit

D1 = Design organized top-down (Boolean)  
D2 = Module dependence (P2/P1)  
D3 = module dependent on prior processing (P3/P1)  
D4 = Database size (P5/P4)  
D5 = Database compartmentalization (P6/P4)  
D6 module single entrance/single exit (P7/P1)

Wi = weighting factor (user assigned) for priority of each issue  
such that sum of Wi = 1

Compute:

DSM = sum over i = 1 to 6 of Wi\*Di

AFSCP: Same calculations. Source provides suggestions on use,  
and weighting factors.

Indicator Name \_\_\_\_\_  
Discovery Metric

Indicator Class (Process or Product)

First Source See SW Requirements Stability

Second Source  
additional sources

Primitives of Indicator

Prime Use of Indicator is:

Additional Uses for Indicator

How Indicator is Computed

Indicator Name \_\_\_\_\_  
Documentation

Indicator Class (Process or Product)

First Source See SW Documentation & Source

Second Source  
additional sources

Primitives of Indicator

Prime Use of Indicator is:

Additional Uses for Indicator

How Indicator is Computed

Indicator Name \_\_\_\_\_  
ESTIMACS

Indicator Class (Process or Product) Process

First Source SEI-CM-12-1.1

Second Source  
additional sources

Primitives of Indicator  
unspecified

Prime Use of Indicator is: Cost

Additional Uses for Indicator  
Schedule, Complexity, Management control

How Indicator is Computed

ESTIMACS is proprietary software that has modules to estimate parameters in the following areas:

System development effort  
Staffing and Costs  
Hardware configuration (estimated equipment required)  
Risk  
Portfolio (estimates effects on development organization)

Indicator Name \_\_\_\_\_  
Error Distributions

Indicator Class (Process or Product) Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
Complete error description

Prime Use of Indicator is: Quality

Additional Uses for Indicator  
Fault, Management Control, Schedule, Cost

How Indicator is Computed

Error description including  
Associated faults  
Types  
Severity  
Phase introduced  
Preventive measure  
Discovery mechanism

Criteria are developed for each classification of error, and errors in each class are counted, and distribution plots are made.

Indicator Name \_\_\_\_\_  
Estimated number Faults remain (seeding)

THIS  
PAGE  
IS  
MISSING  
IN  
ORIGINAL  
DOCUMENT

A-21 & A22

## How Indicator is Computed

$t_i$  = observed times between failures  $i$  and  $i-1$  for a severity level.

$f_i$  = number of failures of a given severity in the  $i$ -th time interval

Source recommends using non-homogeneous Poisson process or Bayesian models to predict reliability  $R(t)$  and cumulative probability functions  $F(t)$ , then plugging them into

$$FR = -1/R(t) * (dR(t)/dt) \quad \text{where } R(t) = 1 - F(t)$$

Indicator Name  
Fault Density

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
F (faults) KSLOC

Prime Use of Indicator is: Quality

Additional Uses for Indicator  
Faults, Reliability, Cost

## How Indicator is Computed

IEEE:

$F$  = faults are the unique faults found over a given time interval for a specific severity class

KSLOC are the source executable and data declarative statements in the code (1000s)

FD is  $F/KSLOC$

AFSCP: Source states metric is similar to defect density, except test data is used instead of inspection data. Inputs



are from test results.

FD = cumulative faults (not failures)/total units in CSCI

Indicator Name \_\_\_\_\_  
Fault-Days Number

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
Phase when fault introduced into system, Date fault introduced,  
Phase/date removed

Prime Use of Indicator is: SW Process

Additional Uses for Indicator  
Faults, Reliability, Quality, Management Control, Cost

How Indicator is Computed

FDi = fault days for the i-th fault.

FDi are summed for FD number at system level, or module level.

Faults are assumed created at middle of phase introduced.

Indicator Name \_\_\_\_\_  
Function Points

Indicator Class (Process or Product) Product

First Source SEI-CM-12-1.1

Second Source

additional sources

Primitives of Indicator

External inputs, output, inquiries, and master files

Prime Use of Indicator is: SW Characteristic

Additional Uses for Indicator

Cost, Complexity

How Indicator is Computed

Weights are assigned as follows:

Inputs, Inquiries = 4

Outputs = 5

Master Files = 10

Weights can be adjusted +/- 35% to accommodate a specific problem's complexity.

After weighting, values are summed.

Indicator Name

Graph-Theoretic Complexity/Architecture

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source

additional sources

Primitives of Indicator

K E N ci

Prime Use of Indicator is: SW Characteristic

Additional Uses for Indicator

Complexity, Maintainability, Cost

## How Indicator is Computed

### Primitives:

K = number of resources

E = number of edges

N = number of nodes

$c_i$  = complexity of module invocation and return along an edge  
(determined by user)

$G_{ki}$  = binary variable to tell if k-th resource is used by i-th edge

$d_k$  = complexity measure for allocation of resource k (user defined)

### Static complexity (nodes and connections) calculations:

$$C = E - N + 1$$

### Generalized static complexity (nodes and resource use by nodes) calculations:

$$C = \sum \text{over all } E \text{ of } (c_i + \sum \text{over all } K \text{ of } (d_k + G_{ki}))$$

### Dynamic complexity (measurement of relative execution time/rate of different modules) calculations:

Use static complexity measure for a given point in time, digesting the data from time average executions or other desired metric

---

### Indicator Name

Incremental Release Content Metric

Indicator Class (Process or Product) Product Process

First Source ESD TR 88-01

Second Source

additional sources

Primitives of Indicator

CSUs

Prime Use of Indicator is: Quality

Additional Uses for Indicator  
Completeness, Rqmts Traceability, Schedule

How Indicator is Computed

Number of CSUs planned for each release is plotted against actuals. Functionality delaying can be spotted, as can unplanned increases in complexity (#s of CSUs).

Indicator Name \_\_\_\_\_  
Independent Process Reliability

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
fij MD Pi qi ri Ri

Prime Use of Indicator is: Reliability

Additional Uses for Indicator

How Indicator is Computed

Process is used to calculate reliability of loopless and database-less systems like TELCO or banking transaction systems.

Primitives:

fij = frequency of execution of transfer from module i to j

MD = # of modules

Pi = i-th process which can be generated in a user environment

qi = probability of Pi

ri = random variable that is 1 if Pi generates correct output and 0 if it does not.

Ri = reliability of i-th module

Model assumes a large quantity of independently designed

implemented and tested modules

then system reliability R is

sum over all i of  $q_i \cdot r_i$

Indicator Name \_\_\_\_\_

Lines of Code

Indicator Class (Process or Product) Product Process

First Source SEI CM 12-1.1

Second Source ESD TR 88-01  
additional sources AFSCP800-43(90)

Primitives of Indicator

Lines of code

total, new, modified, reused, how estimated

Prime Use of Indicator is: SW Characteristic

Additional Uses for Indicator

Cost, Complexity

How Indicator is Computed

SEI:

LOC is identified as most widely used, but also not usable as a predictor (measured after the fact) but can be used to estimate complexity, effort, and productivity.

For a process view, regression analysis has shown that the effort to accomplish a project is

$E = 5.2 \cdot (KLOC)^{.91}$  person-months

ESD TR:

SLOC = source lines of code

For each reporting period, the important factors are:

estimated new SLOC

estimated reused SLOC

estimated modified SLOC

estimated total SLOC

A common definition of SLOC is needed between the contractor and system office.

AFSCP:

Same as ESD TR, added: Estimation method should be tracked.

Indicator Name \_\_\_\_\_

Manhours per Major Defect Detected

Indicator Class (Process or Product) Process

First Source IEEE 982

Second Source

additional sources

Primitives of Indicator

T1 T2 Si I

Prime Use of Indicator is: Cost

Additional Uses for Indicator

Management Control, Cost, Schedule, Manpower

How Indicator is Computed

T1 = prep time for code inspection team

T2 = inspection time for code inspection

Si = number of major (non-trivial) defects found during inspection i

I = total inspections to date

Computed:

$$M = \text{sum over all } I \text{ of } (t_1 + t_2)$$

-----  
sum over all I of Si

Note: computation should start after 8000 lines of code or more have been inspected.

Indicator Name \_\_\_\_\_

Mean time to discover next K faults

Indicator Class (Process or Product) Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
 $f$   $t_i$   $MTTF_i$ (estimated)

Prime Use of Indicator is: Reliability

Additional Uses for Indicator  
Management Control

How Indicator is Computed

$f$  = number of failures found from beginning of testing to present  
 $t_i$  = observed time between  $i-1$  and  $i$ -th failure

$MTTF_{ihat}$  = expected MTTF between  $i$ -th and  $i+1$  fault, as  
estimated by a software model that estimates MTTF values

Mean time to discover next K faults is

= sum of  $t_i$  from  $f$  to  $f+K-1$  of  $MTTF_{ihat}$

Can be use to predict reliability

Indicator Name \_\_\_\_\_  
Mean-Time-To-Failure

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source

additional sources

Primitives of Indicator  
 $t_i$

Prime Use of Indicator is: Reliability

Additional Uses for Indicator  
Quality

How Indicator is Computed

$t_i$  = observed time between  $i$ -th and  $i-1$  failure

All other details are assumed to be common knowledge, and weighting is mentioned as useful in the calculation.

Indicator Name \_\_\_\_\_  
Minimal Unit test case determination

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
N E SN RG

Prime Use of Indicator is: Testing

Additional Uses for Indicator  
Reqmts Traceability, Completeness, Complexity

How Indicator is Computed

Determine cyclomatic complexity of module from complexity metric, then construct test cases to cover all edges of the graph.



Basis Path

Indicator Name \_\_\_\_\_  
Number of Conflicting Requirements

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
inputs outputs functions

Prime Use of Indicator is: Rqmts Traceability

Additional Uses for Indicator  
Reliability, Errors, Completeness, Coverage

How Indicator is Computed

Primitives:  
List of inputs to program  
List of outputs of program  
List of functions performed by program  
Mappings from architecture to requirements are needed.  
Mappings from same spec item to multiple requirements are  
examined for inconsistency  
Mappings from multiple spec items to a single requirement are  
examined for inconsistencies.

Indicator Name \_\_\_\_\_  
Number of entries/exits per module

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
 $E_i$   $X_i$

Prime Use of Indicator is: SW Characteristic

Additional Uses for Indicator  
Complexity, Maintainability

How Indicator is Computed

$E_i$  = entry points for i-th module  
 $X_i$  = exit points fo i-th module

Compute:

$M_i = E_i + X_i$

Indicator Name \_\_\_\_\_  
Open Problem Reports Metric

Indicator Class (Process or Product) Process Product

First Source AFSCP 800-49 (draft)

Second Source  
additional sources

Primitives of Indicator  
Open Problem Reports

Prime Use of Indicator is: SW Process

Additional Uses for Indicator  
Schedule, Cost

## How Indicator is Computed

Use of Pareto charts is recommended to analyze several categorizations of defects:

- 1) By software unit; identifying problem modules.
- 2) By Category: requirements, logic, typo, coding. . .
- 3) By defect discovery method: inspection, peer review, testing. . .
- 4) By line of code
- 5) By severity

2, 3, & 5 are used as measures of the development process. 2 & 3 indicating development problems and 5 suggesting that the management approach may be to delay severe problems until a later date.

Open progress reports are also used in item 51 SW progress, development and test, where SPRs are tracked.

Indicator Name \_\_\_\_\_  
Process Compliance

Indicator Class (Process or Product) Process

First Source SW Quality Indic, Sci Systems

Second Source  
additional sources

Primitives of Indicator  
Compliance with techniques, tools, methods, reviews, reports as  
IDed by SDP

Prime Use of Indicator is: SW Process

Additional Uses for Indicator  
Management Control, Quality

How Indicator is Computed

Monitored by internal review team, and an independent  
government contractor

Indicator Name \_\_\_\_\_  
Process Stability

Indicator Class (Process or Product) Process Product

First Source AFSCP 800-49 (draft)

Second Source  
additional sources

Primitives of Indicator  
Changes in the process broken down by functional area

Prime Use of Indicator is: SW Process

Additional Uses for Indicator  
Cost, Schedule, Reliability, Management Control

How Indicator is Computed

Indicator Name \_\_\_\_\_  
Progress Metric

Indicator Class (Process or Product)

First Source See Design/Unit Completion

Second Source  
additional sources

Primitives of Indicator

Prime Use of Indicator is:

Additional Uses for Indicator

How Indicator is Computed

Indicator Name \_\_\_\_\_  
Quality Evaluation Effort

Indicator Class (Process or Product) Process

First Source SW Qual Indic, Sci Systems Inc

Second Source  
additional sources

Primitives of Indicator  
Percentage of contractor's effort spent on quality evaluation activities

Prime Use of Indicator is: Manpower

Additional Uses for Indicator  
Management Control, Quality, Cost

How Indicator is Computed

Indicator Name \_\_\_\_\_  
Rayleigh Model for Personnel Use

Indicator Class (Process or Product) Process

First Source SEI-CM-12-1.1

Second Source  
additional sources

Primitives of Indicator  
Person-Years for project, KLOC, Technology constant

Prime Use of Indicator is: Manpower

Additional Uses for Indicator  
Cost, Schedule, Management Control

How Indicator is Computed

The method developed was to characterize the personnel use during the development effort. The persons at a time  $t$  is equal to:

$$y = \frac{K \cdot t \cdot (e^{-t^2/2T^2})}{T^2}$$

$K$  = area under Rayleigh curve

$T$  = development time (time of peak staffing requirement)

This is based on a bell curve for personnel usage, starting slow and peaking somewhere approximately 2/3 of the way through the project.

The relationship between size of project and development time was found to be:

$$S = C \cdot (Q^{.3333}) \cdot (T^{1.3333})$$

Where  $S$  = KLOC delivered

$Q$  = effort in person-years

$C$  = state-of-technology constant

Indicator Name \_\_\_\_\_  
Residual Fault Count

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
ti fi

Prime Use of Indicator is: Reliability

Additional Uses for Indicator  
Faults, Risk, Quality

How Indicator is Computed

$t_i$  = observed time interval between  $i$  and  $i-1$  fault  
 $f_i$  = number of failures during  $i$ -th time interval of given severity level.

Calculations are referred to your favorite failure rate class of fault prediction models.

Indicator Name \_\_\_\_\_  
Reliability Growth Function

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
 $NR_k$   $NS$   $nck$

Prime Use of Indicator is: Reliability

Additional Uses for Indicator

How Indicator is Computed

$NR_k$  = number of test cases for the  $k$ -th stage  
 $NS$  = total number of stages  
 $nck$  = total number of successful test cases during  $k$ -th stage

Solve the two simultaneous equations to find  $R(u)$ , reliability

at a particular test period, and A, the growth parameter of the reliability ( $A > 0$  means increasing reliability)

sum over all NS,  $k=1$  to NS for  $(nck/NRk - R(u) + A/k) = 0$

and

sum over all NS,  $k=1$  to NS for  $(nck/NRk - R(u) + A/k)*1/k = 0$

For fault reinsertion rate (bad fix rate) you can use a third equation in conjunction with the first two to find gamma:

sum over all NS,  $k=1$  to NS of:

$(nck/NRk - R(u) + A/(k(1-\gamma)))*1/k(1-\gamma) = 0$

Indicator Name \_\_\_\_\_  
Required Software Reliability

Indicator Class (Process or Product) Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator

Prime Use of Indicator is: Reliability

Additional Uses for Indicator  
Coverage, Risk, Quality, Maintainability

How Indicator is Computed

The entire scheme is stolen from Boehm, Software Engineering Economics with very low to very high ratings in various areas, and can be used to calculate effort required, or reverse-rate an existing product and development

Indicator Name \_\_\_\_\_  
Requirements Compliance



Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
DE N1 N2 N3

Prime Use of Indicator is: Rqmts Traceability

Additional Uses for Indicator  
Quality

How Indicator is Computed

Primitives.

DE = decomposition elements including  
stimulus: external input  
function: defined input/output process  
response: result of the function  
label: numerical identifier for a DE  
reference: spec paragraph number

N are requirements errors found with system verification documents

N1 = errors due to inconsistencies  
N2 = errors due to incompleteness  
N3 = errors due to misinterpretation

Compute:

Decomposition of SRS into DEs

Graph DEs into system verification graph

analyze graph for connectivity and reachability  
Note errors and relative percentages in error classes (N1 N2 N3)

Indicator Name \_\_\_\_\_

## Requirements and Design Stability

Indicator Class (Process or Product) Product

First Source AFSCP 800-43 (90)

Second Source  
additional sources

Primitives of Indicator  
tot rqmts(designs), rqmts(designs) added/deleted/modified SPRs,  
SAIs

Prime Use of Indicator is: Rqmts Traceability

Additional Uses for Indicator  
Rqmts Traceability, Schedule

How Indicator is Computed

Metric is similar to requirements definition and stability  
indicator.

CSCI requirements (designs) are tracked as to baseline, added,  
deleted, and modified. Software Problem Reports (SPRs) and  
Action Items (SAIs) are also tracked (at appropriate phase) as  
to total, number open/closed/unresolved.

Indicator Name \_\_\_\_\_  
Rqmts Definition & Stability/Closure metric

Indicator Class (Process or Product) Product Process

First Source AFSCP 800-43 (86)

Second Source AFSCP 800-49 dr  
additional sources ESD TR 88-01

Primitives of Indicator

SW Rqmts, Unmet Rqmts, Software Size, Open action items, ECPed  
SW units

Prime Use of Indicator is: Rqmts Traceability

Additional Uses for Indicator

Reliability, Coverage, Completeness, Supportability, Cost,  
Schedule

How Indicator is Computed

AFSCP 800-43 (86) This indicator is a combination of the SW Requirements Stability indicator/Discovery Metric (record 3), and the SW Requirements Traceability indicator (record 7) that are referenced in SW Quality indicators and IEEE 982. It also covers material covered in Completeness (record 43), Test Coverage (record 33), Test Coverage, Modular/Functional (record 12), and Defect Removal Rate (record 8).

Indicator looks at requirements traceability, and numbers of Engineering Change Proposals logged against the system. It also compares testing to requirements to ensure that test coverage is adequate and that all requirements are testable. It ensures requirements are traceable up and down the system. The guidance given is that if 15-20% or more of the requirements are not testable the system is considered medium to high risk. The indicator also recommends freezing the requirements for an early delivery increment, with the program being reopened at a later time.

Action items are expected to "spike" up after a review, and decline exponentially after, with good programs having relatively low spikes and rapid resolution of action items.

Indicators are recommended to be collected monthly.

DID references are in source.

AFSCP 800-49:

Speaks to a general correction process. Quantities measured are cumulative open problem reports vs closed reports, and the trend of the two lines to converge (good) or diverge. The indicator states that the difference in the open vs closed reports indicates relative product reliability, and through ease of modifications, supportability.

At an increased difficulty level, 800-49 recommends plotting the changes against time to look for requirements creep, advising

that instability of design can lead to major cost and schedule complications.

ESD TR:

Basically the same as 800-43, the reference recommends measuring requirements changes (additions, deletions, and modifications) and open action items (new and cumulative).

Indicator Name \_\_\_\_\_  
Run Reliability

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
NR nc k S P Pi

Prime Use of Indicator is: Reliability

Additional Uses for Indicator  
Risk

How Indicator is Computed

Primitives:

NR = number of runs made for a given test sample  
nc = number of correct runs for a given test sample  
k = number of runs from a specified period of time  
S = sample space of input patterns/states  
P = probability measure over sample space  
Pi = probability that the i-th run is selected from the sample space (USAGE PROFILE)

Select NR runs randomly IAW Pi, and

Probability of a successful run Pr (estimated) is

$$Pr = \frac{\text{sum of probabilities for correct runs}}{\text{sum of probabilities for all runs}}$$

Run Reliability  $R_k$  for specified period of time and  $k$  runs is

$$R_k = Pr^{**k}$$

Indicator Name \_\_\_\_\_  
SOFTCOST

Indicator Class (Process or Product) Process

First Source SEI-CM-12-1.1

Second Source  
additional sources

Primitives of Indicator  
68 input parameters

Prime Use of Indicator is: Cost

Additional Uses for Indicator

How Indicator is Computed

A questionnaire is administered, and the results of the questionnaire determine the values of the 68 input parameters. Source provides no specifics.

Indicator Name \_\_\_\_\_  
SPQR (SW Productivity, Qual, Reliability)

Indicator Class (Process or Product) Process

First Source SEI-CM-12-1.1

Second Source  
additional sources

Primitives of Indicator  
45 input factors

Prime Use of Indicator is: Cost

Additional Uses for Indicator

How Indicator is Computed

Estimates are based on a response to a 100 question survey on the project, which generates the aforementioned 45 inputs. No specifics are given in source. Model is proprietary.

Indicator Name \_\_\_\_\_  
Software Development Manpower

Indicator Class (Process or Product) Process

First Source AFSCP 800-43 (86)

Second Source ESD TR 88-01  
additional sources

Primitives of Indicator  
Manning Profiles

Prime Use of Indicator is: Manpower

Additional Uses for Indicator  
Schedule, Management Control, Quality

How Indicator is Computed

AFSCP:

The planned manning profiles from the program office and contractor are compared to the actual numbers of personnel, and deviations are noted, both to total personnel quantities and to staff losses.

Frequency of the data collection should be monthly.

Data can be taken per CSCI or per system.

Cautions are mentioned for mythical man-month and high staff turnover situations.

DID references are in source.

ESD TR:

All above applies, plus ratios of experienced to inexperienced personnel are noted. An option is to break out personnel usages by development task.

Indicator Name \_\_\_\_\_  
Software Development Tools

Indicator Class (Process or Product) Process

First Source AFSCP 800-43 (86)

Second Source AFSCP800-43(90)  
additional sources

Primitives of Indicator  
List of Tools, Schedule for tool application, Date tools required

Prime Use of Indicator is: SW Process

Additional Uses for Indicator  
Schedule, Management Control

How Indicator is Computed

(86):

The number computed is the number of months available from when the tools are required to when the tools are expected to be

used.

Data should be collected monthly.

A chart can be developed to track indicator. Negative margins indicate problems, positive margins are desired.

Rules of application of this indicator are to be developed by individual product centers. A laundry list of possible tools required is in source. DID references are in source.

(90): Categories of data on the tools are recommended as follows: Name, Interdependencies (with other tools), Environment (of tool), Purpose, Need Date, Available Date, Status

Indicator Name \_\_\_\_\_  
Software Documentation and Source listings

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source AFSCP 800-14  
additional sources

Primitives of Indicator  
Modularity Descriptiveness Consistency Simplicity Expandability  
Testability

Prime Use of Indicator is: Documentation

Additional Uses for Indicator  
Completeness, Coverage, Maintainability, Rqmts Traceability

How Indicator is Computed

IEEE:  
The source recommends questionnaires to evaluate documentation (AFOTEC Guide), and questionnaires evaluating at least 10% of the source code looking at program readability understandability and maintainability.

AFSCP: Metric named Documentation. Grading scale is recommended 6 point scale, in same areas as IEEE Std.



Explicit computation is

Documentation Index (DI)=

$$\frac{\text{sum over all } i \text{ } w1i*Di/6 + \text{sum over all } i \text{ } w2i*Si/6}{2}$$

Where w1, w2 are weights for documentation and source listings,  
and Di and Si are the ratings for the documents and source  
listings from the reviews.

Indicator Name \_\_\_\_\_  
Software Maturity index

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
Mt Fc Fa Fdel

Prime Use of Indicator is: Quality

Additional Uses for Indicator  
Reliability, Cost Evaluation, SW Characteristic

How Indicator is Computed

Mt = number of modules (functions) in current delivery  
Fc = number of modules (functions) in current delivery that have  
had internal changes from a previous delivery.  
Fa = Number of modules (functions) that are additions to the  
previous delivery  
Fdel = Number of software modules (functions) in the previous  
delivery that have been deleted from the current delivery

Alternate means of calculating:

$$SMI = Mt - (Fa+Fc+Fdel)$$

Mt

or

$$SMI = (Mt - Fc)/Mt$$

Method is dependent on available data

Indicator Name \_\_\_\_\_  
Software Progress--Development and Test

Indicator Class (Process or Product) Process

First Source AFSCP 800-43 (86)

Second Source ESD TR 88-01  
additional sources AFSCP800-43(90)

Primitives of Indicator  
Units, Development of them, tests conducted/passed  
CSCI, SPR

Prime Use of Indicator is: Schedule

Additional Uses for Indicator  
Coverage, Management Control

How Indicator is Computed

AFSCP: This indicator is similar to Design/Unit Completion indicator (record 2).

It takes in primitives of units and the percentage of units 100% designed, percentage of units that have passed testing, and percentage if CSCIs that have been integrated. It also recommends tracking problem reports as they are opened and closed.

A graph of anticipated VS actual for design, test, and integration is recommended for tracking indicators. Guidelines are given for completion percentages between the various activities.  
DID references are in source.

AFSCP 800-43 (90)  
Added FQT progress.

ESD TR:

Primitives:

CSCI = computer software configuration items

SPR = software progress reports

CSC = computer software components

Three calculations are done: Progress of CSCI and systems testing versus planned testing, a graph of cumulative open SPRs versus time  $e$ , and a plot of SPR density per KSLOC. Additional tests may be length of time SPRs are open.

Indicator Name \_\_\_\_\_  
Software Purity level

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
 $t_i$   $f$   $Z_{hat}$

Prime Use of Indicator is: Reliability

Additional Uses for Indicator  
Quality, Risk

How Indicator is Computed

Primitives

$t_i$  = observed times between failures of a given severity level.  
 $f$  = total number of failures in a given interval  
 $Z_{hat}(t)$  = estimated failure rate at (hazard rate) at time  $t$   
(derived from software reliability models)

Compute.

$$PL = \frac{Zhat(t0) - Zhat(tf)}{Zhat(t0)}$$

Indicator Name \_\_\_\_\_  
Software Release Readiness

Indicator Class (Process or Product) Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
Effectiveness measures

Prime Use of Indicator is: Quality

Additional Uses for Indicator  
Risk, Cost

How Indicator is Computed

Measures are assigned values, weighted, and correlated according to risk exposure. Risks are then summed.

Indicator Name \_\_\_\_\_  
Software Requirements Stability

Indicator Class (Process or Product) Process (?)

First Source SW Qual Indic, Sci Sys Inc

Second Source SEI-CM-12-1.1

additional sources AFSCP 800-49 dr

Primitives of Indicator

# Approved "requests for changes" to the SRS or other document

Prime Use of Indicator is: Rqmts Traceability

Additional Uses for Indicator

Cost, Schedule

How Indicator is Computed

for prelim analysis phase, it is changes to the SRS. For detailed design phase, it is changes to top-level spec. In code and test, it is changes to the detailed designs. In all cases, the indicator seems to be the changes found in stage n to the n-1 stage documentation.

SEI speaks to general code changes, and number of design changes.

800-49 talks of problems discovered in code or in baseline documentation. These problems are cumulatively graphed against time, and the slope of the graph is supposed to approach zero (given constant effort), this showing increasing improvement in reliability of the software.

Indicator Name

Software Requirements Traceability

Indicator Class (Process or Product) Process Product

First Source SW Qual Indic, Sci Systems Inc

Second Source IEEE 982

additional sources

Primitives of Indicator

SW Reqmts in SRS, SW reqmts in top level design document IEEE:  
R1,R2

Prime Use of Indicator is: Rqmts Traceability

Additional Uses for Indicator  
Completeness, Coverage

How Indicator is Computed

Indicator is supposed to track linkage between the two requirements realizations. Exact method is unspecified.

IEEE: R1 = number of requirements met by architecture  
R2 = Number of original requirements

Mapping required from SW architecture to original requirements  
to find R1 and R2  
Compute Traceability measure

$TM = 100(R1/R2) \%$

Indicator Name \_\_\_\_\_  
Software Science Measures (Halstead)

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source SEI-CM-12-1.1  
additional sources

Primitives of Indicator  
n1 n2 N1 N2

Prime Use of Indicator is: SW Characteristic

Additional Uses for Indicator  
Complexity, Remaining Faults, Cost

How Indicator is Computed

n1 = unique operators in a program

$n_2$  = unique operands in a program  
 $N_1$  = total occurrences of operators in a program  
 $N_2$  = total operand occurrences in a program

Compute:

Program Vocabulary:  $1 = n_1 + n_2$

Observed program length:  $L = N_1 + N_2$

Estimated program length:  $L = n_1(\log_2(n_1)) + n_2(\log_2(n_2))$

Program volume:  $V = L(\log_2(n_1+n_2))$

Program Difficulty:  $D = (n_1/2)(N_2/n_2)$

Program Level:  $L_1 = 1/D$

Effort:  $E = V/L_1$

Number of Errors:  $B = V/3000 \sim \text{approx } (E^{2/3})/3000$

Time:  $T = E/S$  S taken to be 18 operations/ second

Alternate method for computing length:  $L_{\text{hat}} = \log_2((n_1)!) + \log_2((n_2)!)$

Indicator Name \_\_\_\_\_  
Software Size

Indicator Class (Process or Product)

First Source See "Lines of Code" indicator

Second Source  
additional sources

Primitives of Indicator

Prime Use of Indicator is:

Additional Uses for Indicator

How Indicator is Computed \*

Indicator Name \_\_\_\_\_  
System Performance Reliability

Indicator Class (Process or Product) Product

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator

Prime Use of Indicator is: Reliability

Additional Uses for Indicator  
Quality

How Indicator is Computed

Requirements are established for particular system performance characteristics. Testing is run to determine system load w.r.t. these parameters, and the network "busy" time is calculated. These values are turned into estimations for the actual system efficiency in operation.

Indicator Name \_\_\_\_\_  
Test Accuracy

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator  
Ns Nshat



Prime Use of Indicator is: Testing

Additional Uses for Indicator  
Faults Remaining, Completeness

How Indicator is Computed

Ns = number of seeded faults

Nshat = estimated number of detectable seeded faults

A model is selected to calculate seeded fault detection capability, and the number of seeded faults found at infinity (end of testing) is calculated

$\alpha = Nshat/Ns$

Indicator Name \_\_\_\_\_  
Test Coverage

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source AFSCP 800-14  
additional sources

Primitives of Indicator  
implemented and reqd capabilities, tested and tot program  
primitives(SW Structr)

Prime Use of Indicator is: Testing

Additional Uses for Indicator  
Completeness, Rqmts Traceability, Quality

How Indicator is Computed

Program primitives are broken into functional and data types,  
and are chunked into modules, segments, statements, branches,

nodes or paths.

Data primitives are equivalence classes, Requirements primitives are test cases or functional capabilities.

Recommended implementation:

TC(%) =

$$\frac{\text{implemented capabilities}}{\text{required capabilities}} \times \frac{\text{program primitives tested}}{\text{total program primitives}} \times 100\%$$

AFSCP: Same calculation, except program primitives are called software structure in the equation. Source provides sample ranges for some application classes.

Indicator Name \_\_\_\_\_

Test Progress

Indicator Class (Process or Product)

First Source See- SW Progress, Develop/Test

Second Source  
additional sources

Primitives of Indicator

Prime Use of Indicator is:

Additional Uses for Indicator

How Indicator is Computed

Indicator Name \_\_\_\_\_

Test coverage, Functional or Modular

Indicator Class (Process or Product) Process Product

First Source SW Qual Indic, Sci Systems Inc

Second Source IEEE 982  
additional sources

Primitives of Indicator  
Amount of code exercised by tests IEEE: FE, FT

Prime Use of Indicator is: Testing

Additional Uses for Indicator  
Rqmts Traceability, Schedule, Completeness

How Indicator is Computed

Source recommends using branch coverage measure of test completion, and use of automated tools.

also, for CSC testing, recommends alternate use of percentage of functions tested, or both for that stage.

IEEE:

FE = number of functional (modular) requirements for which all test cases have been satisfactorily completed.

FT = Total number of functional (modular) SW requirements  
computed:

FTC or MTC coverage index:  $FE/FT$

---

Indicator Name  
Testing Sufficiency

Indicator Class (Process or Product) Product Process

First Source IEEE 982

Second Source  
additional sources

Primitives of Indicator

NFhat Fit Fpit Mit Mtot

Prime Use of Indicator is: Testing

Additional Uses for Indicator  
Rqmts Traceability, Quality, Reliability, Coverage

How Indicator is Computed

IEEE:

NFhat = number of faults predicted in software

Fit = total number of faults detected to date in integration testing

Fpit = faults detected prior to integration testing

Mit = modules integrated to date

Mtot = total modules in final configuration

calculate:

$NFrem = (NFhat - Fpit) * Mit / Mtot$

it is recommended that tolerance limits be set 11 and 12 values are recommended .5 and 1.5 respectively such that:

if  $11 * NFrem < Fit < 12 * NFrem$ , testing adequate; if  $Fit > 12 * NFrem$ , check for error-ridden modules; if  $Fit < 11 * NFrem$ , testing may be inadequate (number or variety).

AFSCP: Variables are renamed to: PF=NFhat; FD=Fpit; UT=Mtot; UI=Mit; FR=NFrem; c1, c2=11, 12; MAXT=12\*NFrem, MINT=11\*NFrem. Indicator is the basis for product acceptance. c1 and c2 may be tailored to support experiences.

Indicator Name \_\_\_\_\_  
Tests Completed

Indicator Class (Process or Product) Process

First Source SW Qual Indic, Sci Systems Inc

Second Source  
additional sources

Primitives of Indicator

N

Prime Use of Indicator is: Schedule

Additional Uses for Indicator  
Reliability, Quality

How Indicator is Computed

N = Number of tests completed without a failure.

Similar to record 51-- SW progress, development and test, from  
ESD TR 88-01, AFSCP 800-43 (86)

Indicator Name \_\_\_\_\_  
Time Individual Defects Left Uncorrected

Indicator Class (Process or Product) Process

First Source AFSCP 800-49 (draft)

Second Source  
additional sources

Primitives of Indicator  
Defect Reports

Prime Use of Indicator is: SW Process

Additional Uses for Indicator  
Schedule, Management Control

How Indicator is Computed

cross reference to item 51 -- SW progress, development and  
testing , where time to correct defects as monitored through  
SPRs is mentioned.

## Appendix B Preliminary Data Sheet

CTRL NO:

### PRELIMINARY DATA SHEET FOR INTERVIEWS

NAME: \_\_\_\_\_

We are working with Mr Fred Banks and Min Hue Lu in ENASC to develop a standard set of software indicators for ASD. We were given your name by \_\_\_\_\_ as someone who would be beneficial to us in the research we are doing to develop these indicators. What we need is about 45 minutes of your time to answer some questions about three things: the programs you are currently working, software indicators, and your job. If you are using software indicators, we also request you provide us with a copy of the the software indicator definitions and sources at the time of the interview. We would also like a brief description of the software development process used on your program(s). Any information you provide us will be considered confidential and there will be no ties to specific programs or individuals. Our objective is to develop a composite picture of ASD and not look at specific programs or individuals.

WILL SUPPORT? (Y/N): \_\_\_\_\_

OFFICE SYMBOL: \_\_\_\_\_

PHONE: \_\_\_\_\_

LOCATION: \_\_\_\_\_

MGT OR ENGR: \_\_\_\_\_

COUNTERPART: NAME: \_\_\_\_\_

OFFICE SYMBOL: \_\_\_\_\_

PHONE: \_\_\_\_\_

PROGRAM(S) RESPONSIBLE FOR: \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

CTRL NO:

Software indicators are defined as measured data from the software product and/or the process by which it is developed used to estimate/predict product costs and schedules and to measure productivity and product quality.

ARE YOU CURRENTLY USING SOFTWARE INDICATORS? (Y/N): \_\_\_\_

DATE/TIME OF INTERVIEW: \_\_\_\_\_

NOTES:

---

---

---

---

---

---

---

---

## Appendix C Interview Questionnaires

### Appendix C.1 With Indicators Questionnaire

CTRL NO:

#### QUESTIONNAIRE TO SUPPORT DEVELOPMENT OF STANDARD SOFTWARE MANAGEMENT INDICATORS FOR AERONAUTICAL SYSTEMS DIVISION

**\*\* INDICATORS BEING USED \*\***

#### INTRODUCTION:

We appreciate your participation in the effort to develop a standard set of software management indicators for ASD. Your inputs are an important part of our research effort and the key to its success. As we have previously stated, we are working with ASD/ENASC in support of their avionics database. One thing we want to make absolutely clear up-front is that we are only interested in obtaining a global ASD perspective of the software management process. As such, each individual input we receive will be considered confidential, and there will be **no ties** to specific programs or individuals. We will gladly share with you the results of our project if you would like us to do so.

The survey consists of three parts. The first part asks questions about general information concerning your program and software development effort. The second part asks questions about software indicators. The third part asks questions about your job and information needs. The entire survey should take no more than 45 minutes. If you have no questions, let's begin.

**Note:** If an answer to a question is unknown, ask if there is an individual who may be able to answer it.



CTRL NO:

PART I: GENERAL INFORMATION

1. Verify the program(s) they are responsible for based on Preliminary Data Sheet.
2. A. When was the contract awarded for your program(s)?  
B. When will the contract end for your program(s)?
3. What type of contract is it (i.e., FFP, CPIF)?
4. Is cost performance reporting required for your contract?
5. What System life cycle phase is your program(s) currently in?
6. What is the estimated size of the software (LOC)?

CTRL NO:

PART II: CURRENT INDICATOR ENVIRONMENT

7. What do you believe are the primary reasons why you are currently using software indicators (please, no more than 5)?

8. Are these indicators required by contract? YES / NO

A. If NO, what do you believe are the primary reasons why they were not placed on contract?

B. If YES, how did you motivate the contractor to accept the task of providing software indicator data?

9. Please describe the process that was followed to come up with the software indicators being used?

**CTRL NO:**

10. Please describe the process by which software indicator data is gathered?

11. What happens to the information provided by the software indicators?

12. Have the software indicators significantly influenced:

A. Your ability to accomplish your job? YES / NO  
Why / Why not?

B. The program? YES / NO                      Why / Why not?

**CTRL NO:**

13. \*\* Hand Software Indicator Sheet to Individual to Fill Out \*\*

14. \*\* Hand Software Indicator Publications Sheet to Individual to Fill Out \*\*

15. Have you had any changes to the software indicators you are using? YES / NO

A. If YES, why were the changes made (please, no more than 5 reasons)?

16. Do you expect any future changes to the software indicators you are using? YES / NO

A. IF NO: Why not?

B. IF YES:

1) Why will changes be made (please, no more than 5 reasons)?

2) What software indicators do you envision being used in the future? Please include sources.

CTRL NO:

PART III: JOB AND INFORMATION NEEDS

17. How many years have you been involved with software development efforts?

18. What types of training have you had to prepare you for your job, both formal and informal?

19. How long have you been in your current job?

20. Please briefly describe your current job (responsibilities, functions performed, etc.)?

21. \*\* Hand Software Development Areas Sheet to Individual to Fill Out \*\*

22. What types of information do you currently get to help you accomplish your job?

**CTRL NO:**

23. How does this information differ from what you actually need to help you accomplish your job?

24. What types of information products do you produce to provide information to higher management?

25. Is there anything else you feel is important to the use of software indicators that we have not covered?

26. What are your overall impressions of the questionnaire we just asked you?

CTRL NO:

OTHER POSSIBLE PEOPLE TO INTERVIEW?

NAME

OFFC SYMBOL

PHONE


FOLLOW-UP ACTIONS THAT NEED TO BE ACCOMPLISHED:

- ☐ Have list of software indicators and sources
- ☐ Have description of software development process


CTRL NO:

SOFTWARE INDICATOR SHEET - PART I

Please review these software indicators and indicate the following:

- a. Y - I AM familiar with it, N - I AM NOT familiar with it.
  - b. If there are indicators that you are familiar with that are not listed here, please identify those and their source if you know it.
  - c. For those software indicators that you put a Y by or identified, please rate them based on the scale below in terms of how important they are to accomplishing your job:
  - d. For those software indicators that you rated a 5 or 3 or less please provide a brief statement as to why you rated it that way:
- e.x. #15 - Documentation you rate a 5, beneath it you put "documentation is my way of seeing what the contractor is doing"

5	4	3	2	1
Very				Very
Important	Important	Neutral	Unimportant	Unimportant

1. \_\_\_ Computer Resource Utilization
2. \_\_\_ Fault Density
3. \_\_\_ Software Development Manpower
4. \_\_\_ Test Coverage
5. \_\_\_ Testing Sufficiency
6. \_\_\_ Cost/Schedule Deviations
7. \_\_\_ Completeness
8. \_\_\_ Requirements Definition and Design Stability
9. \_\_\_ Schedule Progress



CTRL NO:

SOFTWARE INDICATOR SHEET - PART II

- |          | 5  | 4         | 3       | 2           | 1           |
|----------|--|-----------|---------|-------------|-------------|
|          | Very                                       |           |         |             | Very        |
|          | Important                                  | Important | Neutral | Unimportant | Unimportant |
| 10. ____ | Defect Density                             |           |         |             |             |
| 11. ____ | Memory Utilization                         |           |         |             |             |
| 12. ____ | Software Documentation and Source Listings |           |         |             |             |
| 13. ____ | Throughput                                 |           |         |             |             |
| 14. ____ | Requirements Traceability                  |           |         |             |             |
| 15. ____ | Documentation                              |           |         |             |             |
| 16. ____ | Deviation of End Product...                |           |         |             |             |
| 17. ____ | Mean-Time-To-Failure                       |           |         |             |             |
| 18. ____ | Requirements Compliance                    |           |         |             |             |
| 19. ____ | Software Progress - Development and Test   |           |         |             |             |
| 20. ____ | Defect Distribution                        |           |         |             |             |
| 21. ____ | Test Accuracy                              |           |         |             |             |
| 22. ____ | Software Size                              |           |         |             |             |
| 23. ____ | Design Progress                            |           |         |             |             |

Using the Scale above, what do you believe is the importance of using Software Indicators in software development efforts? \_\_\_\_

CTRL NO:

**SOFTWARE INDICATOR PUBLICATIONS SHEET**

Please review the following software indicator publications and indicate: Y - I HAVE read it, N - I HAVE NOT read it.

- ☐ AFSCP 800-43, Software Management Indicators
- ☐ IEEE Std 982, IEE Std Dictionary of Measures to Produce Reliable Software
- ☐ ESD-TR-88-01, Software Management Metrics
- ☐ AFSCP 800-14, Software Quality Indicators
- ☐ SEI-CM-12-1.1, Software Metrics
- ☐ AFSCP 800-49, Software Quality Indicators (DRAFT)
- ☐ DODD 5000.2, Defense Acquisition Program Procedures
- ☐ AMC-P 70-13, Software Management Indicators

If there are other software indicator publications that you have read that are not listed above, please list them:

CTRL NO:

**SOFTWARE DEVELOPMENT AREAS SHEET**

Please review the following areas of software development and rate them based on the scale below in terms of how important they are to **doing** your job:

5	4	3	2	1
Very Important	Important	Neutral	Unimportant	Very Unimportant

1.   \_\_\_   Schedule/Project Tracking
2.   \_\_\_   Reliability
3.   \_\_\_   Cost Projection/Estimation/Tracking
4.   \_\_\_   Software Performance (i.e., throughput, memory utilization)
5.   \_\_\_   Software Characteristics (i.e., size, function points, complexity)
6.   \_\_\_   Quality
7.   \_\_\_   Manpower
8.   \_\_\_   Testing
9.   \_\_\_   Requirements Traceability/Stability
10.  \_\_\_   Maintainability
11.  \_\_\_   Documentation
12.  \_\_\_   Software Process Consistency

Now, from the list above, please identify the **TOP 5** areas that you feel are critical to **doing** your job and list them below:

---

---

---

---

---

CTRL NO:

**QUESTIONNAIRE TO SUPPORT DEVELOPMENT OF  
STANDARD SOFTWARE MANAGEMENT INDICATORS  
FOR AERONAUTICAL SYSTEMS DIVISION**

**\*\* NO INDICATORS BEING USED \*\***

**INTRODUCTION:**

We appreciate your participation in the effort to develop a standard set of software management indicators for ASD. Your inputs are an important part of our research effort and the key to its success. As we have previously stated, we are working with ASD/ENASC in support of their avionics database. One thing we want to make absolutely clear up-front is that we are only interested in obtaining a global ASD perspective of the software management process. As such, each individual input we receive will be considered confidential, and there will be **no ties** to specific programs or individuals. We will gladly share with you the results of our project if you would like us to do so.

The survey consists of three parts. The first part asks questions about general information concerning your program and software development effort. The second part asks questions about software indicators. The third part asks questions about your job and information needs. The entire survey should take no more than 45 minutes. If you have no questions, let's begin.

**Note:** If an answer to a question is unknown, ask if there is an individual who may be able to answer it.

**CTRL NO:**

**PART I: GENERAL INFORMATION**

1. Verify the program(s) they are responsible for based on Preliminary Data Sheet.
2. A. When was the contract awarded for your program(s)?  
B. When will the contract end for your program(s)?
3. What type of contract is it (i.e., FFP, CPIF)?
4. Is cost performance reporting required for your contract?
5. What System life cycle phase is your program(s) currently in?
6. What is the estimated size of the software (LOC)?

CTRL NO:

PART II: CURRENT INDICATOR ENVIRONMENT

7. What do you believe are the primary reasons why software indicators are not currently being used (please, no more than 5)?

8. How do you estimate/predict costs, schedules, productivity, and product quality?

9. \*\* Hand Software Indicator Sheet to Individual to Fill Out \*\*

**CTRL NO:**

10. \*\* Hand Software Indicator Publication Sheet to Individual  
to Fill Out \*\*

11. Do you envision software indicators being used on your  
program(s) in the future? YES / NO

A. Why / Why not (please, no more than 5 reasons)?

B. If YES, what software indicators do you envision being  
used? Please include sources.

CTRL NO:

PART III: JOB AND INFORMATION NEEDS

12. How many years have you been involved with software development efforts?

13. What types of training have you had to prepare you for your job, both formal and informal?

14. How long have you been in your current job?

15. Please briefly describe your current job (responsibilities, functions performed, etc.)?

16. \*\* Hand Software Development Areas Sheet to Individual to Fill Out \*\*

17. What types of information do you currently get to help you accomplish your job?



**CTRL NO:**

18. How does this information differ from what you actually need to help you accomplish your job?

19. What types of information products do you produce to provide information to higher management?

20. Is there anything else you feel is important to the use of software indicators that we have not covered?

21. What are your overall impressions of the questionnaire we just asked you?

CTRL NO:

OTHER POSSIBLE PEOPLE TO INTERVIEW?

NAME

OFFC SYMBOL

PHONE


FOLLOW-UP ACTIONS THAT NEED TO BE ACCOMPLISHED:

☐

Have description of software development process


CTRL NO:

SOFTWARE INDICATOR SHEET - PART I

Please review these software indicators and indicate the following:

a. Y - I AM familiar with it, N - I AM NOT familiar with it.

b. If there are indicators that you are familiar with that are not listed here, please identify those and their source if you know it.

c. For those software indicators that you put a Y by or identified, please rate them based on the scale below in terms of how important they are to accomplishing your job:

d. For those software indicators that you rated a 5 or 3 or less please provide a brief statement as to why you rated it that way:

e.x. #15 - Documentation you rate a 5, beneath it you put "documentation is my way of seeing what the contractor is doing"

5	4	3	2	1
Very Important	Important	Neutral	Unimportant	Very Unimportant

1. \_\_\_ Computer Resource Utilization

2. \_\_\_ Fault Density

3. \_\_\_ Software Development Manpower

4. \_\_\_ Test Coverage

5. \_\_\_ Testing Sufficiency

6. \_\_\_ Cost/Schedule Deviations

7. \_\_\_ Completeness

8. \_\_\_ Requirements Definition and Design Stability

9. \_\_\_ Schedule Progress

CTRL NO:

SOFTWARE INDICATOR SHEET - PART II

5	4	3	2	1
Very Important	Important	Neutral	Unimportant	Very Unimportant
10. ____	Defect Density			
11. ____	Memory Utilization			
12. ____	Software Documentation and Source Listings			
13. ____	Throughput			
14. ____	Requirements Traceability			
15. ____	Documentation			
16. ____	Deviation of End Product...			
17. ____	Mean-Time-To-Failure			
18. ____	Requirements Compliance			
19. ____	Software Progress - Development and Test			
20. ____	Defect Distribution			
21. ____	Test Accuracy			
22. ____	Software Size			
23. ____	Design Progress			

Using the Scale above, what do you believe is the importance of using Software Indicators in software development efforts? \_\_\_\_

CTRL NO:

SOFTWARE INDICATOR PUBLICATIONS SHEET

Please review the following software indicator publications and indicate: Y - I HAVE read it, N - I HAVE NOT read it.

- \_\_\_ AFSCP 800-43, Software Management Indicators
- \_\_\_ IEEE Std 982, IEE Std Dictionary of Measures to Produce  
Reliable Software
- \_\_\_ ESD-TR-88-01, Software Management Metrics
- \_\_\_ AFSCP 800-14, Software Quality Indicators
- \_\_\_ SEI-CM-12-1.1, Software Metrics
- \_\_\_ AFSCP 800-49, Software Quality Indicators (DRAFT)
- \_\_\_ DODD 5000.2, Defense Acquisition Program Procedures
- \_\_\_ AMC-P 70-13, Software Management Indicators

If there are other software indicator publications that you have read that are not listed above, please list them:

CTRL NO:

### SOFTWARE DEVELOPMENT AREAS SHEET

Please review the following areas of software development and rate them based on the scale below in terms of how important they are to **doing** your job:

5 Very Important	4 Important	3 Neutral	2 Unimportant	1 Very Unimportant
------------------------	----------------	--------------	------------------	--------------------------

1.   \_\_\_   Schedule/Project Tracking
2.   \_\_\_   Reliability
3.   \_\_\_   Cost Projection/Estimation/Tracking
4.   \_\_\_   Software Performance (i.e., throughput, memory utilization)
5.   \_\_\_   Software Characteristics (i.e., size, function points, complexity)
6.   \_\_\_   Quality
7.   \_\_\_   Manpower
8.   \_\_\_   Testing
9.   \_\_\_   Requirements Traceability/Stability
10.  \_\_\_   Maintainability
11.  \_\_\_   Documentation
12.  \_\_\_   Software Process Consistency

Now, from the list above, please identify the **TOP 5** areas that you feel are critical to **doing** your job and list them below:

---

---

---

---

---

### Appendix C.3 ASC/ENASC Questionnaire

WITH: \_\_\_\_\_

#### ASD/ENASC QUESTIONNAIRE

1. Who originated the database effort?
2. When was it originated?
3. Why was it originated?
4. What are the primary objectives of the database?
5. Is there any formal documentation identifying the need for the database, validating it as a requirement, or providing tasking for it? YES / NO If YES, can we get a copy?
6. Who do you foresee as the "customer" of the database effort?
7. What do you believe the customer needs from the database?

8. What do you expect to provide the customer?

9. How do you expect to provide it?

**FOLLOW-UP ACTIONS THAT NEED TO BE ACCOMPLISHED:**

☐ Have formal documentation for database

---

---

---

---

---

---

---

---

---

---



## ENASC SURVEY

Please review the following areas of software development and rate them based on the scale below in terms of how significant they are to the database:

1	2	3	4	5
Very				Very
Significant	Significant	Neutral	Unsignificant	Unsignificant

- \_\_\_ Schedule/Project Tracking
- \_\_\_ Reliability
- \_\_\_ Cost Projection/Estimation/Tracking
- \_\_\_ Software Performance
- \_\_\_ Quality
- \_\_\_ Manpower
- \_\_\_ Testing
- \_\_\_ Requirements Traceability/Stability
- \_\_\_ Maintainability
- \_\_\_ Documentation

Appendix C.4 Database Customer Questionnaire

TELEPHONE INTERVIEW OF THE USERS OF THE ENASC DATABASE

NAME: \_\_\_\_\_ MGR/ENGR? \_\_\_\_\_

OFFICE SYMBOL: \_\_\_\_\_

When did you use the ENASC database?

What stage of the program were you in?

What were your specific objectives/ needs in using the database?

Were these objectives/needs satisfied? Why/Why not?

Is there anything you would like to see added to/ changed in the database?

Would you use the database again? Why/Why not?

Appendix D: Survey Responses  
Appendix D.1: With Indicators Survey Responses

This database has been sorted by question number. Not all interviewees were asked all questions due to survey revisions (response normally left blank). In addition, not all interviewees responded to every question asked. The first six questions were programmatic data and have been grouped at the beginning of the report.

Control # 4

CONTR AWARD 87	CONTRACT END 93
CONTRACT TYPE: FFP IF	COST PERF REPORT?
PROG PHASE EMD (testing)	EST SIZE

Control # 5

CONTR AWARD 91 March	CONTRACT END 96 (EMD)
CONTRACT TYPE: CPIF (award fee)	COST PERF REPORT?
PROG PHASE EMD (restart)	EST SIZE

Control # 6

CONTR AWARD 92	CONTRACT END 97
CONTRACT TYPE: CPIF (award fee?)	COST PERF REPORT?
PROG PHASE EMD	EST SIZE

Control # 8

CONTR AWARD 91 Mar	CONTRACT END 96 Mar
CONTRACT TYPE: CPIF (award fee)	COST PERF REPORT?
PROG PHASE EMD (prelim design done)	EST SIZE

Control # 9A

CONTR AWARD 83	CONTRACT END IOC 93, P3I->95
CONTRACT TYPE: basic FFP	COST PERF REPORT?
PROG PHASE code production/delivery	EST SIZE

Control # 10

CONTR AWARD 88	CONTRACT END 94
CONTRACT TYPE: CP	COST PERF REPORT?
PROG PHASE late EMD/early production	EST SIZE

Control # 12A

CONTR AWARD 91	CONTRACT END 99
CONTRACT TYPE: CPAF	COST PERF REPORT?
PROG PHASE EMD	EST SIZE

Control # 13

CONTR AWARD 88	CONTRACT END 93
CONTRACT TYPE: FFP	COST PERF REPORT?
PROG PHASE EMD	EST SIZE

Control # 14

CONTR AWARD 91	CONTRACT END 99 (EMD ends)
CONTRACT TYPE: CP	COST PERF REPORT? YES
PROG PHASE EMD	EST SIZE 70K tot (40K new)

Control # 15

CONTR AWARD 91	CONTRACT END 99
CONTRACT TYPE: CP	COST PERF REPORT? YES
PROG PHASE EMD	EST SIZE 1 MLOC,+120K COTS

Control # 16

CONTR AWARD 79 (83 production)	CONTRACT END 89 (still going)
CONTRACT TYPE: CP	COST PERF REPORT? YES
PROG PHASE Production (complete)	EST SIZE 1MLOC+10*~10K programs

Control # 17

CONTR AWARD 88	CONTRACT END 94
CONTRACT TYPE: FFP	COST PERF REPORT? ???
PROG PHASE EMD	EST SIZE don't know

Control # 18

CONTR AWARD 87	CONTRACT END 95
CONTRACT TYPE: FFP	COST PERF REPORT? ??-no
PROG PHASE EMD	EST SIZE 50KLOC inc firmware

Control # 19

CONTR AWARD 89	CONTRACT END 92
CONTRACT TYPE: FFP	COST PERF REPORT? NO
PROG PHASE EMD done, mod exist A/C EST SIZE 250 KLOC total	

Control # 20

CONTR AWARD 91	CONTRACT END 2002, dev-94
CONTRACT TYPE: CP IF	COST PERF REPORT? YES
PROG PHASE EMD	EST SIZE 1.2M BYTES

Control # 21

CONTR AWARD 90	CONTRACT END 94
CONTRACT TYPE: FFP	COST PERF REPORT? NO
PROG PHASE EMD (completed CDR) EST SIZE > 75K	

Control # 23

CONTR AWARD 91 ????	CONTRACT END 94
CONTRACT TYPE: CP AF	COST PERF REPORT? YES
PROG PHASE EMD	EST SIZE 1 to 2 MLOC

Control # 24

CONTR AWARD 83                      CONTRACT END sched 96 93/4??  
CONTRACT TYPE: FFP                      COST PERF REPORT? YES  
PROG PHASE concurrent production   EST SIZE 20KLOC flt SW

Control # 25

CONTR AWARD 88-on 81 contr    CONTRACT END 95  
CONTRACT TYPE: FFP w/CP ECP    COST PERF REPORT? YES  
PROG PHASE EMD/prodn portions of   EST SIZE 800KLOC (400K COTS)

Control # 26

CONTR AWARD 91                      CONTRACT END 2000  
CONTRACT TYPE: CPAF                      COST PERF REPORT? YES  
PROG PHASE EMD                      EST SIZE 500KLOC

Control # 27

CONTR AWARD 83 dev, 85 prod    CONTRACT END extend to 93  
CONTRACT TYPE: FFP                      COST PERF REPORT? unkwn  
PROG PHASE production    EST SIZE 1024k words (16 bit)

Control # 28

CONTR AWARD 89(ph1), ph2-91    CONTRACT END 94  
CONTRACT TYPE: CPIF                      COST PERF REPORT? YES  
PROG PHASE EMD/concurrent production   EST SIZE 5-7 MLOC

Control # 30

CONTR AWARD 81                      CONTRACT END 94  
CONTRACT TYPE: FFP                      COST PERF REPORT? YES (no->SW)  
PROG PHASE EMD                      EST SIZE 640KLOC

Control # 31

CONTR AWARD 82

CONTRACT END 94

CONTRACT TYPE: FFP

COST PERF REPORT? YES-sys cost

PROG PHASE EMD

EST SIZE unknown

Control # 9B

CONTR AWARD 83

CONTRACT END 94 IOC

CONTRACT TYPE: FFP

COST PERF REPORT?

PROG PHASE production/delivery EST SIZE

Control # 12B

CONTR AWARD 91

CONTRACT END 99

CONTRACT TYPE: CPAF

COST PERF REPORT?

PROG PHASE EMD

EST SIZE

QUESTION 7

Control # 4

WHAT ARE REASONS FOR INDICATORS?

provides status to top-level management for the following areas: tells us status of contractor effort, schedule; memory and throughput; provides work completion status for the contractor management hierarchy

Control # 5

WHAT ARE REASONS FOR INDICATORS?

status of where development effort is at, assists in verifying that contractor is correctly applying resources/efforts

Control # 6

WHAT ARE REASONS FOR INDICATORS?

insight into software development process and progress, never had it before

Control # 8

WHAT ARE REASONS FOR INDICATORS?

contractor progress; look for trends; MIL-STD 1815; AFR compliance (RFP review process)

Control # 9A

WHAT ARE REASONS FOR INDICATORS?

monitor contractor progress/compliance with specification

Control # 10

WHAT ARE REASONS FOR INDICATORS?

SW is "nebulous", indicators help quantify where the program is at in development cycle

-----  
Control # 12A

WHAT ARE REASONS FOR INDICATORS?

right thing to do; need to know what's going on; have to plan where you are at; program/plan tracking

-----  
Control # 13

WHAT ARE REASONS FOR INDICATORS?

unsure of reason (contractor)?

-----  
Control # 14

WHAT ARE REASONS FOR INDICATORS?

visibility to make decisions; spot problem areas -> memory, throughput, breach thresholds; look at trends -> \*right info at right time to make decisions

-----  
Control # 15

WHAT ARE REASONS FOR INDICATORS?

indicators used to be proactive, not reactive; to be able to manage problem not have problem manage you

-----  
Control # 16

WHAT ARE REASONS FOR INDICATORS?

indicators were used to determine how complete SW was prior to testing; track SPRs during unit development

-----  
Control # 17

WHAT ARE REASONS FOR INDICATORS?

metrics are being provided informally to the manager, but not widely being used

-----  
Control # 18

WHAT ARE REASONS FOR INDICATORS?

updates to keep track of "where software is going"

-----  
Control # 19

WHAT ARE REASONS FOR INDICATORS?

track development progress; reviews audits test-contractor ready for it; ready for milestones; find out problems; size metric important early on for 1/3 new code Ada waiver; answers all questions about SW development (air staff inquiries)

-----  
Control # 20

WHAT ARE REASONS FOR INDICATORS?

to keep track of program, performance; to see if contractor met requirements; there is a (higher management) \_program requirement\_ for the contractor to have indicators; another avenue of getting data

-----  
Control # 21



WHAT ARE REASONS FOR INDICATORS?  
to determine whether contractor is performing to SOW and contract

---

Control # 23

WHAT ARE REASONS FOR INDICATORS?  
there were problems keeping the contractor on schedule, recently  
implemented management indicators; replanning effort

---

Control # 24

WHAT ARE REASONS FOR INDICATORS?  
trend analysis (SPR tracking, are problems related?) -> cost purposes ->  
evaluate contractor schedule; requirements analysis; accounting for  
efficiency; evaluate contractor proposed value; hard to use REVIC

---

Control # 25

WHAT ARE REASONS FOR INDICATORS?  
monitor progress; form a baseline to compare milestones to; manage cost  
and manpower and complexity answer the question "is the task too diffi-  
cult?"

---

Control # 26

WHAT ARE REASONS FOR INDICATORS?  
SW development integrity program--compliance with MIL-STD 1803

---

Control # 27

WHAT ARE REASONS FOR INDICATORS?  
money--development is expensive, and we want to be sure we are getting a  
sound product (value); user input, meeting the customer's needs

---

Control # 28

WHAT ARE REASONS FOR INDICATORS?  
insight into development process, with a system of this size, this is  
vital cost control, does program meet targets (work performed versus  
cost to date); quality

---

Control # 30

WHAT ARE REASONS FOR INDICATORS?  
project management--track SW development progress; memory, throughput,  
timing and SLOC estimates are done to force contractor to perform  
estimation task (would have preferred monthly or quarterly formal  
estimates)

---

Control # 31

WHAT ARE REASONS FOR INDICATORS?  
validate schedule, cost reporting; provide management insight/oversight

---

Control # 9B

WHAT ARE REASONS FOR INDICATORS?  
monitor contractor progress/compliance with spec

---

Control # 12B

WHAT ARE REASONS FOR INDICATORS?

right thing to do; need to know what's going on; have to plan where you are at; program/plan tracking

QUESTION 8

Control # 4

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

recommended early on to have indicators on contract, but contractor wanted exorbitant amount to put them on; several other attempts have been made to put them on contract, most recent was 3 years ago (cost still determinant); indicators would seem to provide no leverage with contractor as he is losing money (not sure what to do with them in this case); contractor not able/willing to make product improvement efforts

-----  
Control # 5

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

-----  
Control # 6

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

-----  
Control # 8

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

-----  
Control # 9A

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

no guidance to do so at the time; no training highlighting the need for indicators

-----  
Control # 10

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

contractor realized there was a need for indicators, and accepted them

-----  
Control # 12A

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

no reply

-----  
Control # 13

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

-----  
Control # 14

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

contractor meetings; made indicators part of integrated master plan (system engineering master plan); metrics were used by contractor and SPO; MIL-STD 1803

-----  
Control # 15

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

award fee basis for indicator effort; integrity; based on successful completion

-----  
Control # 16

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

Navy DIDs were used on the program, as the program started out as a Navy development

-----  
Control # 17

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

no emphasis when contracted

-----  
Control # 18

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

unknown--follow up contact given

-----  
Control # 19

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

no problems before contract was awarded, however, after contract award, prime contractor was only integrator, and subcontractors did not want to information; bad fight-- "tooth and nail"; much negotiation; initial indicators were very bad -> expectation line kept changing, "earned value" definition used; management not too excited

-----  
Control # 20

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

there was the initial question as to why the SPO wanted them, but the contractor's desire was to satisfy the customer (SPO); no problem

-----  
Control # 21

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

no problem

-----  
Control # 23

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

small group of people working on program; have a responsible contractor; streamlining; constant contact between SPO and contractor (good communication); "nature" of program

-----  
Control # 24

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

"nature" of program and time frame; "getting a product that works is what counts" mentality (time pressure); organic support/ maintenance not considered

-----  
Control # 25

INDICATORS REQUIRED BY CONTRACT ? NO(?)

WHAT ARE PRIMARY REASONS WHY NOT?

probably not on contract, as the method they were selected wouldn't seem to be a way to get them on contract (see Q9)

-----  
Control # 26

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

not that they're heard of; SDP; none known

-----  
Control # 27

INDICATORS REQUIRED BY CONTRACT ? YES ?

HOW DID YOU MOTIVATE CONTRACTOR?

performance measures are on contract, but usability ones are not; contractor agreed to use indicators fairly easily, amenable

-----  
Control # 28

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

contractor already believed indicators were helpful to them, and so were willing to use them

-----  
Control # 30

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

indicators "did not exist" at time of contract award, and were not put on at a later time because doing so was cost prohibitive, and the contractor was providing some indicators "free" but sporadically

-----  
Control # 31

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

contract is old (early 80s) and indicators weren't the thing to do at that time; instead, SPO got whatever the contractor was using

-----  
Control # 9B

INDICATORS REQUIRED BY CONTRACT ? NO

WHAT ARE PRIMARY REASONS WHY NOT?

no guidance to do so at time; no training highlighting the need for indicators

-----  
Control # 12B

INDICATORS REQUIRED BY CONTRACT ? YES

HOW DID YOU MOTIVATE CONTRACTOR?

no reply

-----

#### QUESTION 9

Control # 4

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

in the RFP, AFSC 800-43 was sent out as suggested indicators, contractor responded with subset of those, and these indicators are generally followed by the contractor

-----

Control # 5

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

-----

Control # 6

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

AFSCP 800-43 (tailored, based on what he wanted); TQ CAT team and contractor effort resulted in SW development process manuals (2167 based) as the contractor's internal indicator model; these two were combined for indicators for development

-----

Control # 8

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

contractor proposed indicators, negotiations between contractor and SPO

-----

Control # 9A

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

presented by contractor; progress report on high level specification;

-----

Control # 10

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

started with AFSCP 800-43, had study effort between SPO and contractor to fine tune indicators; reevaluations as program progresses to continue fine tuning process

-----

Control # 12A

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

dem/val, constantly harped on up need for indicators, evaluated demonstration metrics and acclimated contractor to them during dem/val; tried experimental data measures, data sets, frequencies; selected correct metrics per phase; iterated process; don't look at lines of code "get what you measure"; acquired indicators by contractor

-----

Control # 13

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

contractor proposed indicators

-----

Control # 14

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

(also see Q8B) ASD(CR) pamphlet was used as starting point; contractor meetings ironed out which metrics; SCCR identified risk areas

-----

Control # 15

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

SPO proposed set, contractor worked with SPO and eventually concurred with set; SDCCRs and MIL-STD 1803s -> worked with contractor teams to

get best possible metrics; made sure SDP contained indicator information

-----  
Control # 16

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS  
-----

Control # 17

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

contractor internal indicators are being "used"  
-----

Control # 18

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

may be part of 2167 indicators having to be used -> came from a management indicator guide (unsure of name); former engineer had a CRAC course; joint informal agreement between contractor and SPO  
-----

Control # 19

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

used all of AFSCP 800-43 (86) except cost (not privy to information); added size indicator to determine need for Ada waiver; basically SPO self-decided  
-----

Control # 20

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

program policy to use indicators; started with ENFZ standard DID (DI-MCCR-80459), had discussion with contractor to iron out details  
-----

Control # 21

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

don't know--was not on program at that time  
-----

Control # 23

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

did literature search to see what other organizations were doing; evolutionary approach, take a set that works for someone else and adapted them; SPO specifically wanted schedule, process, progress, and quality indicators--beyond these, it was left to the contractor to identify actual indicators to start with, and the contractor evolves them; did not mandate which indicators, because this leads to no motivation on the contractor's part to use them--"useless for us to force a set of indicators on a contractor, he must adapt to his own system"  
-----

Control # 24

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

selected by product improvement; \*experience in other programs\* ->gain insight into the types of issues that come up and what information you need to help identify those issues; categorize problem and specify how long a turn-around time is wanted (solving problems)  
-----

Control # 25

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS

current set of indicators resulted from an EN tiger team recommendation--indicators were already on being used, but EN recommend-

ed that more direct data be used in charts, and to add SPR tracking and milestone tracking

-----  
Control # 26

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS  
no; no at this point

-----  
Control # 27

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS  
interviewee did not know

-----  
Control # 28

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS  
indicators used were the "standard set" used within the SPO

-----  
Control # 30

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS  
the contractor used AFSCP 800-43 as a basis, and levied those indicator requirements on their suppliers, and came up with their own DIDs for them

-----  
Control # 31

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS  
SPO examined the indicators that the contractor was using, and where they were unsuitable, asked the contractor to use industry "standard" indicators (AFSCP 800-43, Software Engineering Economics)

-----  
Control # 9B

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS  
presented by contractor; progress report on high level specification; show use of 2167

-----  
Control # 12B

DESCRIBE PROCESS FOLLOWED TO COME UP WITH INDICATORS  
during dem/val phase, constantly brought up the need for indicators, evaluated demonstration metrics during dem/val; tried experimental data measures, data sets, frequencies; selected correct metrics per phase; iterated process; don't look at lines of code "get what you measure"; acquired indicators by contractor by ???

-----  
QUESTION 10

-----  
Control # 4

PROCESS BY WHICH INDICATOR DATA IS GATHERED?  
contractor has weekly meeting that contains information -> software report, contractor puts this information together and forwards it to SPO; also gather information from major meeting's minutes

-----  
Control # 5

PROCESS BY WHICH INDICATOR DATA IS GATHERED?  
most data reported in software status report (monthly), a "snapshot" of contractor database

-----  
Control # 6

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

(will be) monthly reports of digested data; AFSCP 800-43 -> combined report data item and metric data item that is what the contractor is doing

-----

Control # 8

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

monthly SW status report, includes current status and trends; SW development accounting document; cadre teamwork; copy of contractor database (monthly)

-----

Control # 9A

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

delivered at PMRs (PDR, CDR, etc); meeting minutes; telephone conversations

-----

Control # 10

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

contractor analyzes indicators and prepares report; quarterly TCM meetings the indicators are gone over; reports generated every other month, or monthly during critical stages of development; fine tuning is done by reporting for each release

-----

Control # 12A

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

contractor updates the following two systems: SW engineering environment and management technical information system (has TPMs on it), and SPO personnel have access to these systems; format was worked out with contractor, includes high level management indicators that are "rolled up" from TPMs; metrics are a subset of the overall program TPMs

-----

Control # 13

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

contractor generates data after CSU is complete

-----

Control # 14

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

contractor generates a monthly report; SPO also has access to the primitives, and does an independent analysis to verify contractor report; home office runs models to cross-check

-----

Control # 15

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

"product team" environment, joint effort to gather information together; program status is briefed at each review, PSR, IPR; problems and strong points are both highlighted at meetings

-----

Control # 16

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

important problems warranted a phone call, less important ones



"non-critical" were briefed at PMRs

-----  
Control # 17

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

contractor provides monthly updates via reports and also computer updates

-----  
Control # 18

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

contractor provides report

-----  
Control # 19

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

data provided monthly as part of status report; contractor generated; engineers eventually stopped getting reports

-----  
Control # 20

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

data is gathered from contractor database (tied to SW engineering tools)  
-> tied to compiler; also, there are measurements available on a similar development that had somewhat different specifications

-----  
Control # 21

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

indicator data is gathered by CDRL items, driven by events

-----  
Control # 23

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

contractor generates data from time cards, interviews with managers; read data carefully, includes assessment where they are relative to progress and assessment of impacts and management reserve; weekly telecon with contractor to explain report; IFC (project leaders) rotate visits to plant

-----  
Control # 24

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

indicators are gathered through counterpart at contractor; have a good working relationship; counterpart passes on informally what SPO asks for; weekly or biweekly conference calls are conducted with SPO, user, maintainer, and contractor to pass along information; component level "produceability" data was being gathered, data got better later in development

-----  
Control # 25

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

SPR data and milestone data is provided to SPO directly (weekly); contractor also analyzes data by department and tracks SPR closures internally

-----  
Control # 26

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

"metrics" tool by DEC has latest copy of SW eng environment tools,

collection will done automatically; SW development status report (quarterly) contains indicators in the technical performance measures, also one of higher-level and reported differently

-----

Control # 27

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

for the initial part of development and testing, the contractor provides all information, when the Air Force begins testing, then the SPO begins to get raw data from tests (joint effort to analyze)

-----

Control # 28

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

raw data was processed by contractor who provides a report at a semi-monthly PMR, in addition raw data is delivered to SPO for verification, if desired

-----

Control # 30

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

contractor provided a report of the computed indicators, and SPO performed some trend analysis on the reports

-----

Control # 31

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

indicators were provided to SPO in both reports containing progress charts (graphs) and the "raw data", tabular listings of the data for each module, and the explanations for each (SUDS reports?)

-----

Control # 9B

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

delivered at PMRs (PDR, CDR); meeting minutes; telephone conversations

-----

Control # 12B

PROCESS BY WHICH INDICATOR DATA IS GATHERED?

contractor updates the following two systems; SW engineering environment and management technical information system that has TPM s on it, and SPO personnel have access to the se systems format was worked out with contractor; includes high level management indicators that are "rolled up"

from TPMs; metrics are a subset of the overall program TPMs

QUESTION 11

-----

Control # 4

WHAT HAPPENS TO INDICATOR DATA?

send information to manager and higher-level engineers; provide verbal status

-----

Control # 5

WHAT HAPPENS TO INDICATOR DATA?

engineering and program office review status report and software development accounting document (providing look at entire database using automated analysis tools), accounting document is a unique DID that is

provided on tape from the contractor

-----  
Control # 6

WHAT HAPPENS TO INDICATOR DATA?

(will be) reviewing indicators, (wants to) do trend analysis on several months of reports (needs guidance)

-----  
Control # 8

WHAT HAPPENS TO INDICATOR DATA?

SW team report to program manager; track contractor progress; generate actions if behind (schedule); prove to ALC, users, Air Staff that SW is progressing

-----  
Control # 9A

WHAT HAPPENS TO INDICATOR DATA?

used as a signal to indicate when a meeting is required; identifies missing requirements

-----  
Control # 10

WHAT HAPPENS TO INDICATOR DATA?

look at indicators, hammer on contractor when numbers don't look right; indicators provide early identification of problems so corrective action can be applied

-----  
Control # 12A

WHAT HAPPENS TO INDICATOR DATA?

"rule the world"; certain metrics are "rolled up" into a "front office group" that are briefed to program manager (agreed upon by PM); action item database is used to track documents

-----  
Control # 13

WHAT HAPPENS TO INDICATOR DATA?

helps contractor design CSUs, but too late for the SPO to do anything about information

-----  
Control # 14

WHAT HAPPENS TO INDICATOR DATA?

get monthly report from contractor -> compare SW estimates to HW capability-> look at requirements and verify that they can still be met (examine reserve capacity requirements)

-----  
Control # 15

WHAT HAPPENS TO INDICATOR DATA?

dissect information and determine what the indicator is saying action depends on information provided--if the information is "bad", then apply appropriate management resources to solve problems; good information increases award fee, bad information decreases award fee

-----  
Control # 16

WHAT HAPPENS TO INDICATOR DATA?

problems were prioritized for funding purposes, and priority levels had to be established (fixable or not, mission critical or nuisance)

-----  
Control # 17

WHAT HAPPENS TO INDICATOR DATA?

indicators were tried initially because of the team concept in SPO -> didn't work (non-responsive); indicators are distributed to flight test, engineers, managers for their information; interviewee uses the indicators to try to draw trends, to go to the contractor and find out what's going on

-----  
Control # 18

WHAT HAPPENS TO INDICATOR DATA?

sometimes use it for in-house memos as a "heads-up"; tool to keep SPO informed, spotlight risks and SW status; contractor uses indicators internally; problems highlighted by indicators can lead to TIMs to find solutions

-----  
Control # 19

WHAT HAPPENS TO INDICATOR DATA?

because the indicators were not part of a DID, management got the indicators, and it was on their whim to pass the information on to the engineers, unless the engineers asked for the information; also because the indicators were not on a DID, there was trouble finding anything to "report them against"; contractor wanted a waiver, but engineer held ground and refused until contractor provided data, example of leverage

-----  
Control # 20

WHAT HAPPENS TO INDICATOR DATA?

some indicators go to higher management, they are rolled into higher level indicators; also combined into the "executive level" software indicator that is briefed; another use is to track program

-----  
Control # 21

WHAT HAPPENS TO INDICATOR DATA?

data comes in for evaluation and approval/disapproval; disapproval leads to negotiation to resolve conflicts

-----  
Control # 23

WHAT HAPPENS TO INDICATOR DATA?

indicators come into project leaders (IFC) and they disseminate them to OFCs, if any issues need to be resolved, they contact contractor counterpart; basically, use tools (indicators) as a means to identify when \_management\_ needs to get \_involved\_; identify problems

-----  
Control # 24

WHAT HAPPENS TO INDICATOR DATA?

indicator data passed on to management and chief engineer; used in weekly/biweekly briefings to managers, team meetings where the impacts are discussed; data is analyzed for slips in schedule

-----  
Control # 25

WHAT HAPPENS TO INDICATOR DATA?

verifies development is meeting contract requirements; provides a heads up if a deviation is present (time, manpower), allows SPO to ask for more information on what the deviation is; informs upper level management

-----  
Control # 26

WHAT HAPPENS TO INDICATOR DATA?

interviewee's job is to manage indicator collection system, individual teams will use indicators in their own way, depending on their own needs

-----  
Control # 27

WHAT HAPPENS TO INDICATOR DATA?

indicator information is analyzed to see if a correction is required, and if it is, SPO works with contractor to get the problem fixed; when the schedule is in jeopardy, the problem is elevated to higher levels; SPO works with user to determine if slip is tolerable--time criticality of a release being fielded versus a later correction having to be done

-----  
Control # 28

WHAT HAPPENS TO INDICATOR DATA?

award fee is based on performance, and indicators are written into the award fee criteria; \*\* indicators are also used as guidelines to identify problems and possible take appropriate action

-----  
Control # 30

WHAT HAPPENS TO INDICATOR DATA?

trend analysis was done by SPO, and indicator information was briefed weekly to SPO director during times that were critical to the SW development effort

-----  
Control # 31

WHAT HAPPENS TO INDICATOR DATA?

indicators were used to predict schedules and costs

-----  
Control # 9B

WHAT HAPPENS TO INDICATOR DATA?

used as a signal to indicate when a meeting is required; identifies missing requirements

-----  
Control # 12B

WHAT HAPPENS TO INDICATOR DATA?

"rule the world" certain metrics are "rolled up" into a "front office group" that are briefed to program manager (agreed upon by PM); action item database is used to track

-----  
QUESTION 12

-----  
Control # 4

INDICATORS INFLUENCED ABILITY TO DO JOB ? NO

The indicators helped, provided insight into process and how they are derived, but they provide more internal influence to contractor

INDICATORS INFLUENCED THE PROGRAM ? NO

(same as 12A) helped from the point of view of the contractor, but not done much for the program office except give status

-----  
Control # 5

INDICATORS INFLUENCED ABILITY TO DO JOB ? Y/N

the indicators are all geared toward the code and test phases of the program, and are not useful in the current phase (requirements analysis), as they do not provide much information; too early to tell utility later in program

INDICATORS INFLUENCED THE PROGRAM ? Y/N

same as 12A  
-----

Control # 6

INDICATORS INFLUENCED ABILITY TO DO JOB ? N/A

no data yet

INDICATORS INFLUENCED THE PROGRAM ? N/A

no data yet  
-----

Control # 8

INDICATORS INFLUENCED ABILITY TO DO JOB ? YES

provided good insight into program status

INDICATORS INFLUENCED THE PROGRAM ? YES

program managers/engineers/ALCs show concern over indicator data  
-----

Control # 9A

INDICATORS INFLUENCED ABILITY TO DO JOB ? YES

helps watch specification requirements; helps crosscheck proposal

INDICATORS INFLUENCED THE PROGRAM ? NO

software is being developed in a good manner, indicators of less importance on "well running" program  
-----

Control # 10

INDICATORS INFLUENCED ABILITY TO DO JOB ? YES

insight provided to program management office and contractor, as contractor uses same set of indicators internally

INDICATORS INFLUENCED THE PROGRAM ? YES

keeps program on schedule; catches problems early enough to keep release(??) on schedule  
-----

Control # 12A

INDICATORS INFLUENCED ABILITY TO DO JOB ? YES

having a plan/following a plan is important; plan data shows where you are at--deviations "jump off the page"; large software efforts behave statistically

INDICATORS INFLUENCED THE PROGRAM ? YES

check designs against what is predicted -> weight problem  
-----

Control # 13

INDICATORS INFLUENCED ABILITY TO DO JOB ? NO/Ys

too late for the SPO to do anything about indicator by the time it is

presented; it does help contractor to build better code, though  
INDICATORS INFLUENCED THE PROGRAM ? YES  
indicator made the code better  
-----

Control # 14  
INDICATORS INFLUENCED ABILITY TO DO JOB ? NO  
indicators are a tool that definitely helps, but is not at a significant level (yet), at least at this stage of the game (haven't been used much yet); there is some value added by the indicators; indicators are not a panacea  
INDICATORS INFLUENCED THE PROGRAM ? YES  
visibility -> indicators can show inconsistencies between different \_areas\_; indicators provide monthly reports of system levels  
-----

Control # 15  
INDICATORS INFLUENCED ABILITY TO DO JOB ? YES  
provide team agreement on when things occur; control budget and schedule, helps in analyzing these things; also helps because job description now blurred between management and engineering, given more "management flavor" in engineering  
INDICATORS INFLUENCED THE PROGRAM ? YES  
indicators provide visibility in ways that the SPO did not have before; more emphasis on systems, on the path to do avionics development from the start  
-----

Control # 16  
INDICATORS INFLUENCED ABILITY TO DO JOB ? YES  
  
kept current status of development; how complete program was  
INDICATORS INFLUENCED THE PROGRAM ? YES  
indicators became part of record of SW, when the system was PMTR'ed, the assuming authority knew the condition of the SW; if not tracked (?) then, would not have been worthwhile to manage  
-----

Control # 17  
INDICATORS INFLUENCED ABILITY TO DO JOB ? YES  
visibility -> tells about what program is and how it is going; positive vs. negative; "what job is all about"--helps foresee problems  
INDICATORS INFLUENCED THE PROGRAM ? NO  
not being used by the program as a whole  
-----

Control # 18  
INDICATORS INFLUENCED ABILITY TO DO JOB ? YES  
structured approach to getting information, vice haphazard gathering; already laid out what information you are getting  
INDICATORS INFLUENCED THE PROGRAM ? YES  
helps engineer be prepared to do her job; a prepared engineer benefits the program  
-----

Control # 19  
INDICATORS INFLUENCED ABILITY TO DO JOB ? YES  
provided insight; could have gotten more use if the indicators were put

on contract correctly, and management participated ; AFOTEC is using data to figure maturity and ?? data; would have been nice to know ahead of time \*\*\*SW maturity

INDICATORS INFLUENCED THE PROGRAM ? NO

nobody listened, briefed in briefing to higher ups, management didn't use them; implementation bad, how to use them and how to read them

-----  
Control # 20

INDICATORS INFLUENCED ABILITY TO DO JOB ? NO

can get data that indicators provide from other sources

INDICATORS INFLUENCED THE PROGRAM ? NO

can get data provided from other sources; SW is a small piece of the overall effort, lower priority

-----  
Control # 21

INDICATORS INFLUENCED ABILITY TO DO JOB ? YES

gives insight into what is going on with contractor, whether or not contractor is on track (meeting milestones); insight into contractor thought process

INDICATORS INFLUENCED THE PROGRAM ? YES

same as 12A, insight into contractor

-----  
Control # 23

INDICATORS INFLUENCED ABILITY TO DO JOB ? YES

much better ability to quantify schedule activities--"better idea when coming into a window"

INDICATORS INFLUENCED THE PROGRAM ? YES

same as 12a, quantifiability

-----  
Control # 24

INDICATORS INFLUENCED ABILITY TO DO JOB ? YES

indicators give insight, so the SPO knows when slips are going to come ahead of time and look for alternatives, increased visibility; can use data as stick against contractor

INDICATORS INFLUENCED THE PROGRAM ? YES

same as 12a, warn of slips and provide visibility

-----  
Control # 25

INDICATORS INFLUENCED ABILITY TO DO JOB ? NO

interviewee's job is not affected, as job is at a lower level, and does not involve interpreting indicator data

INDICATORS INFLUENCED THE PROGRAM ? YES

indicators provide visibility; "warm fuzzy" that things are on track; reduced slippage; forced contractor to think through activities and schedule requirements and provide better estimates

-----  
Control # 26

-----  
Control # 27

INDICATORS INFLUENCED ABILITY TO DO JOB ? YES

indicators are the key to defining whether or not SPO is getting product



that was ordered, test to see if the product is usable  
INDICATORS INFLUENCED THE PROGRAM ? YES  
product correctness and usability are the overall program drivers;  
indicators define what goes into a product  
-----

Control # 28  
INDICATORS INFLUENCED ABILITY TO DO JOB ? YES  
indicators are the things the program manager looks at and it gets their  
attention, gets them interested, focuses action  
INDICATORS INFLUENCED THE PROGRAM ? YES  
see Q12a; focuses attention on problems  
-----

Control # 30  
INDICATORS INFLUENCED ABILITY TO DO JOB ? NO  
management indicators (schedule) were strictly used by management  
(interviewee is engineer); there are no indicators being collected that  
cover technical areas; management and engineering have clearly defined  
responsibilities; engineering could use indicators such as size and  
throughput to estimate manhours required (an engineering function)  
INDICATORS INFLUENCED THE PROGRAM ? YES  
indicators forced contractor to establish plan, and allowed SPO to "nail  
him" (contractor) to come up with corrective actions when the indicators  
showed that the contractor was not meeting plans  
-----

Control # 31  
INDICATORS INFLUENCED ABILITY TO DO JOB ? YES  
indicators are an absolute necessity, the only insight into what pro-  
gress is being made on SW development; indicators are basic management  
information applied (focused) to SW developments  
INDICATORS INFLUENCED THE PROGRAM ? YES  
SPO director was briefed weekly on indicator information, and after  
having been briefed the SPO director would have direct conversations  
with the company president each week  
-----

Control # 9B  
INDICATORS INFLUENCED ABILITY TO DO JOB ? YES  
help watch specification requirements; helps crosscheck proposal  
INDICATORS INFLUENCED THE PROGRAM ? NO  
software is being developed in a good manner, indicators of less impor-  
tance on "well running" programs  
-----

Control # 12B  
INDICATORS INFLUENCED ABILITY TO DO JOB ? YES  
having a plan/following a plan is important; plan data shows where you  
are at--deviations "jump off the page" large software efforts behave  
statistically  
INDICATORS INFLUENCED THE PROGRAM ? YES  
check designs against what is predicted -> weight problem  
-----

#### QUESTION 15

-----

Control # 4

ANY CHANGES TO INDICATORS ? YES

WHY ?

contractor does not want to keep up with the effort to collect indicators (costs); contractor does not want to report downward trends

-----

Control # 5

ANY CHANGES TO INDICATORS ? NO

-----

Control # 6

ANY CHANGES TO INDICATORS ? NO

-----

Control # 8

ANY CHANGES TO INDICATORS ? YES

WHY ?

changes made to improve presentation of metric, no new metrics have been introduced

-----

Control # 9A

ANY CHANGES TO INDICATORS ? NO

-----

Control # 10

ANY CHANGES TO INDICATORS ? YES

WHY ?

fine tune indicators, add more visibility (functionality builds broken out for individual release completeness); deleted some indicators (requirements volatility, test discrepancies) because they were providing no value added; when get into low level, can use raw data; also, indicators could be wrong level (mostly top-level)

-----

Control # 12A

ANY CHANGES TO INDICATORS ?

-----

Control # 13

ANY CHANGES TO INDICATORS ? NO

-----

Control # 14

ANY CHANGES TO INDICATORS ? YES

WHY ?

tweaking to create better visibility, modifying current set, especially visibility into ratios of high-order language to assembler

-----

Control # 15

ANY CHANGES TO INDICATORS ? NO

-----

Control # 16

ANY CHANGES TO INDICATORS ? NO

-----

Control # 17

ANY CHANGES TO INDICATORS ? YES

WHY ?

changes were a more precise definition of the terms "open" and "resolved"; also a better indication of what should be tracked

-----  
Control # 18

ANY CHANGES TO INDICATORS ? NO  
-----

Control # 19

ANY CHANGES TO INDICATORS ? YES

WHY ?

got rid of requirements stability one, contractor convinced management would not change -> driven by short program  
-----

Control # 20

ANY CHANGES TO INDICATORS ? YES

WHY ?

cut back from original 30 indicators to approximately 10, as the software indicators were in greater numbers than rest of program, far higher percentage of total indicators than importance warranted; some indicators cut back to indicating minor timing cycles only (less detail)  
-----

Control # 21

ANY CHANGES TO INDICATORS ? NO  
-----

Control # 23

ANY CHANGES TO INDICATORS ? YES

WHY ?

indicators were not initially useful, were not telling what really wanted -> missing or incomplete information; looked at "perfective" changes; data added to provide completeness; needed the right data to do functions like updating planning parameters, estimates on the project, and historical data  
-----

Control # 24

ANY CHANGES TO INDICATORS ? NO  
-----

Control # 25

ANY CHANGES TO INDICATORS ? YES

WHY ?

EN team's recommendations added chart-ability to existing indicator data and provided more confidence in \_schedule\_; improved contractor's projections  
-----

Control # 26

ANY CHANGES TO INDICATORS ? YES

WHY ?

if an indicator is not meaningful, drop it, get something else to meet the needs of a particular team or functional area; also tailoring for reporting frequency; interviewee's job is to set up a standard system and let teams using that system customize it  
-----

Control # 27

ANY CHANGES TO INDICATORS ? YES

WHY ?

started by looking at requirements only--a specific combination of parameters--now broaden definition to "combat viability" to include a full operational usefulness

-----  
Control # 28

ANY CHANGES TO INDICATORS ? NO

-----  
Control # 30

ANY CHANGES TO INDICATORS ? YES

WHY ?

changed detail and method collection (changed weights on project completion sub-tasks) to better reflect progress in the project, and avoid discontinuities (jumps) in the charts tracking percentage complete in the project

-----  
Control # 31

ANY CHANGES TO INDICATORS ? YES

WHY ?

the graphs that were used to predict project progress did not reflect the performance that was obtained, therefore the "linear" predictions were replaced with a better fitting "curve" projection

-----  
Control # 9B

ANY CHANGES TO INDICATORS ? NO

-----  
Control # 12B

ANY CHANGES TO INDICATORS ?

-----  
QUESTION 16

-----  
Control # 4

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

late stage of program; contractor does not believe in taking indicator data on faults during testing, which is where the effort is concentrated now

-----  
Control # 5

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

changes will be modifications to existing indicators there have been misunderstandings as to what indicators are going to be delivered under the contract (contractor is concerned about proprietary data); investigation is being done as to why one requested indicator is not being reported "fallen through the cracks"

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

no reply

-----  
Control # 6

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

too early to tell, there may end up being changes made

-----  
Control # 8

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

improve visibility into program as the program moves into the coding/testing phase

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

not sure  
-----

Control # 9A

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

program is going well  
-----

Control # 10

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

too late in program  
-----

Control # 12A

DO YOU EXPECT FUTURE CHANGES ?  
-----

Control # 13

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

change of complexity number from a requirement to a goal, to reduce the need for deviations

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

no reply  
-----

Control # 14

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

continue tweaking to provide more visibility

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

no reply  
-----

Control # 15

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

continue to evolve and adapt to changes in new requirements

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

rely on experience, expertise, and knowledge of teams to come up with new indicators  
-----

Control # 16

DO YOU EXPECT FUTURE CHANGES ?  
-----

Control # 17

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

to further improve on the indicator -> evolution

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

no reply

-----  
Control # 18

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

indicator list is considered "thorough" already

-----  
Control # 19

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

too short a program, no time left

-----  
Control # 20

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

current indicator set adequate

-----  
Control # 21

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

not sure, unable to predict "crystal ball stuff, Murphy's law"

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

no reply

-----  
Control # 23

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

\*\* when you don't know how to approach a problem, build things parametrically, and indicators give better parametric data to go back and update planning parameters; looking to do SEI process assessment on team, therefore the contractor is looking to expand knowledge and expertise in metrics and indicators

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

more evolution of existing metrics versus generating new ones

-----  
Control # 24

DO YOU EXPECT FUTURE CHANGES ? NO(?)

WHY NOT ?

hard to say if there would be any changes; more like this is what interviewee would like to have, versus actual change

-----  
Control # 25

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

two full time contractor personnel are looking at indicators, contractor is now excited about indicators and should be continuing to fine tune indicators

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

see 16b1, contractor generated changes

-----  
Control # 26

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

see Q15a; system will be tailored by each team to meet their needs

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

anticipated areas of change for teams would be development phase-dependent indicators being used and set aside as the development progresses

-----  
Control # 27

DO YOU EXPECT FUTURE CHANGES ? YES

WHY ?

scope of indicator definition may narrow due to new supporting agency (ALC)

instead of contractor, and their definitions are of a narrow scope

WHAT INDICATORS DO YOU ENVISION, AND SOURCES ?

no reply

-----  
Control # 28

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

very successful with set currently being used

-----  
Control # 30

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

project development is about complete

-----  
Control # 31

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

program is finished

-----  
Control # 9B

DO YOU EXPECT FUTURE CHANGES ? NO

WHY NOT ?

program is going well

-----  
Control # 12B

DO YOU EXPECT FUTURE CHANGES ?

-----  
QUESTION 17/18

Control # 4

YEARS INVOLVED WITH SOFTWARE ? 7 yr

WHAT TYPES OF TRAINING HAVE YOU HAD ?

BS CS; CRAC; Ada, 1553, 2167, cost estimating classes at AFIT

-----  
Control # 5

YEARS INVOLVED WITH SOFTWARE ? 2 yr

WHAT TYPES OF TRAINING HAVE YOU HAD ?

2167/2168 courses; BSEE; sys 100/200; AFIT general acquisition short courses

Control # 6

YEARS INVOLVED WITH SOFTWARE ? 1.5 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
CRAC; SAS-006; 2167/2168 seminar; MSEE

-----

Control # 8

YEARS INVOLVED WITH SOFTWARE ? 2.5 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
CRAC; BS comp eng; sys100; graduate SW development courses at WSU;  
contractor provided training on ADARTS - SPC

-----

Control # 9A

YEARS INVOLVED WITH SOFTWARE ? 12 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
A: OJT; BSEE

-----

Control # 10

YEARS INVOLVED WITH SOFTWARE ? 15 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
BS EE; AFIT short courses (Ada, microprocessor, computer project  
management); tri-Ada conference

-----

Control # 12A

YEARS INVOLVED WITH SOFTWARE ? 18 yr (a)  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
A: OJT; courses in languages (jovial, Ada); AFIT course in computer  
resources; microprocessors course; 2167/2168 seminar

-----

Control # 13

YEARS INVOLVED WITH SOFTWARE ? 2 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
BS computer engineering; OJT (primary means)

-----

Control # 14

YEARS INVOLVED WITH SOFTWARE ? 9 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
BSEE; CRAC; SW engineering courses at AFIT; OJT/prior technical jobs at  
Wright labs; language courses at WSU

-----

Control # 15

YEARS INVOLVED WITH SOFTWARE ? 7 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
BSEE; CRAC; AFIT and EN seminars; OJT and prior civilian job experience  
working in software quality

-----

Control # 16

YEARS INVOLVED WITH SOFTWARE ? 8 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
BSEE; CRAC; undergraduate computer language courses; a lot of OJT

-----

Control # 17

YEARS INVOLVED WITH SOFTWARE ? 3 months



WHAT TYPES OF TRAINING HAVE YOU HAD ?  
none.

-----  
Control # 18

YEARS INVOLVED WITH SOFTWARE ? 1.5 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
CRAC; intro to MCCR management, SQA, and SW cost estimation class from  
AFIT; contractor provided SW requirements traceability course  
-----

Control # 19

YEARS INVOLVED WITH SOFTWARE ? 10 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
CRAC; sys100,200; BS aeronautics; systems management course for computer  
systems  
-----

Control # 20

YEARS INVOLVED WITH SOFTWARE ? 9 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
OJT in support systems; BSEE; sys100; ENE training (sys100 for ATE, HW &  
SW development process); OJT at present position  
-----

Control # 21

YEARS INVOLVED WITH SOFTWARE ? 5 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
CRAC; computer courses; OJT, self-study  
-----

Control # 23

YEARS INVOLVED WITH SOFTWARE ? 11 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
OJT primarily; BSEE (computer design); informal courses with software  
-----

Control # 24

YEARS INVOLVED WITH SOFTWARE ? 11 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
BSEE, computer background; microprocessor software experience; sys100;  
AFIT master's courses; 2167/2168 training; programming courses  
-----

Control # 25

YEARS INVOLVED WITH SOFTWARE ? 3 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
primarily OJT (large amount); BSEE; CRAC; sys100; AFIT SW engineering  
class  
-----

Control # 26

YEARS INVOLVED WITH SOFTWARE ? 7 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
BS CS; AFIT SW eng classes; language training; sys100; 2167 training  
-----

Control # 27

YEARS INVOLVED WITH SOFTWARE ? 1.5 yr  
WHAT TYPES OF TRAINING HAVE YOU HAD ?  
management classes; primarily OJT

-----  
Control # 28

YEARS INVOLVED WITH SOFTWARE ? 15 yr

WHAT TYPES OF TRAINING HAVE YOU HAD ?

BS CS; MCC course; AFIT sys engineering, basic computer architecture ,  
and SW cost estimation courses; DSMC-management of SW acquisition; CRAC;  
2167 course; SW engineering with Ada course  
-----

Control # 30

YEARS INVOLVED WITH SOFTWARE ? 15 yr

WHAT TYPES OF TRAINING HAVE YOU HAD ?

CRAC course; AFIT SW engineering courses; BSEE  
-----

Control # 31

YEARS INVOLVED WITH SOFTWARE ? 22 yr

WHAT TYPES OF TRAINING HAVE YOU HAD ?

MS in management; DSMC course--management of SW acquisition  
-----

Control # 9B

YEARS INVOLVED WITH SOFTWARE ? 8 yr (b)

WHAT TYPES OF TRAINING HAVE YOU HAD ?

BSEE; AFIT PCE courses on SW development; short courses on fortran,  
digital controls; uP development  
-----

Control # 12B

YEARS INVOLVED WITH SOFTWARE ? 9 yr (b)

WHAT TYPES OF TRAINING HAVE YOU HAD ?

BSEE; ATC ada course; sys 400; primarily OJT (prior job experience);  
DSMC long course; SDCCR  
-----

~~QUESTION-19/20~~-----

Control # 4

YEARS IN CURRENT JOB ? 6 yr

DESCRIBE JOB

documentation review; interpret 2167 requirements versus contractor  
performance, evaluate deviations; evaluate manpower, completeness of  
work; evaluate software testing (adequacy, meeting requirements); inter-  
facing with HW engineers; evaluate legal aspects of software  
-----

Control # 5

YEARS IN CURRENT JOB ? 3 yr

DESCRIBE JOB

monitor cost, schedule, and performance of software development effort;  
work with program control, engineering, and scheduling functions  
-----

Control # 6

YEARS IN CURRENT JOB ? 1.5 yr

DESCRIBE JOB

hardware, software, systems engineer for system; lead engineer -> tech-  
nical requirements of system; deals with requirements, documentation,  
testing  
-----

Control # 8

YEARS IN CURRENT JOB ? 2 yr

DESCRIBE JOB

lead SW engineer; coordinate reviews (SDP, indicators, etc); review contractor documentation; item captain-source selection

-----

Control # 9A

YEARS IN CURRENT JOB ? 2 yr

DESCRIBE JOB

advisor; comments on software

-----

Control # 10

YEARS IN CURRENT JOB ? 3 yr

DESCRIBE JOB

avionics SW integrator; supervises 3-4 people; requirements definition, documentation, contract management and monitoring; work with AF test agencies on test plans, both test labs and flight testing agencies

-----

Control # 12A

YEARS IN CURRENT JOB ? 4 yr (a)

DESCRIBE JOB

lead of the avionics team; responsible for cost, schedule, produceability

-----

Control # 13

YEARS IN CURRENT JOB ? 2 yr

DESCRIBE JOB

SW engineer for program; coordinate with other services and contractor; document review; SW testing review, witness

-----

Control # 14

YEARS IN CURRENT JOB ? 6 yr

DESCRIBE JOB

validation and verification of contractor's efforts through documentation/specification (including SRS) review; guidance to contractor on interpretation of requirements, standards, and creation/ implementation of program policies; keep program manager up to date on SW development effort, identify problem areas

-----

Control # 15

YEARS IN CURRENT JOB ? 2 yr

DESCRIBE JOB

responsible for product; managing cost, schedule, delivery, and quality of product to customer

-----

Control # 16

YEARS IN CURRENT JOB ? 8 yr

DESCRIBE JOB

electronic support equipment support primary responsibility; system software; reviewing documentation; not present at tests, no formal sell-off to government; maintainability of software -> ensuring appropriate documentation

-----  
Control # 17

YEARS IN CURRENT JOB ? 6 months

DESCRIBE JOB

tracking requirements -> justify requirements; ECP evaluation; monitor indicators

-----  
Control # 18

YEARS IN CURRENT JOB ? 1.5 yr

DESCRIBE JOB

review all software documentation; lead group of people doing SW evaluation; participate in SW review; provide technical input to program office on status of SW

-----  
Control # 19

YEARS IN CURRENT JOB ? 2.5 yr

DESCRIBE JOB

mission critical computer resources focal point for 2-letter; all engineering SW responsibilities for SPO; integration engineer on particular type of systems (responsible for all SW)

-----  
Control # 20

YEARS IN CURRENT JOB ? 4 yr

DESCRIBE JOB

responsible for SW technology (not logic)--system will have larger degree of SW control than similar systems previously procured, how to use SW for this control; use of Ada and expert systems

-----  
Control # 21

YEARS IN CURRENT JOB ? 5 yr

DESCRIBE JOB

engineering functional part of the project; work closely with the contractor; elucidate specifications and design for contractor; making sure contractor fulfills specification requirements and what those requirements are; project engineer

-----  
Control # 23

YEARS IN CURRENT JOB ? 5 yr

DESCRIBE JOB

deputy for IFC development team; responsible for all airborne software; monitor tactical groups working IFCs, do strategic planning; long range planning of OFP and contents of releases; more management type things (interviewee is engineer)

-----  
Control # 24

YEARS IN CURRENT JOB ? 2 yr

DESCRIBE JOB

responsible for flight software and mission computer--design development and testing of SW; help supporting command set up support facility

-----  
Control # 25

YEARS IN CURRENT JOB ? 2.5 yr

DESCRIBE JOB

computer security; SW test responsibility (organization and execution);  
working future requirements

-----  
Control # 26

YEARS IN CURRENT JOB ? 1 yr

DESCRIBE JOB

team leader; participates in core processing group (system central  
computer); document reviews; participates in working groups for tools  
refinement; interfaces with contractor to work out differences between  
product and need; computer resources focal point

-----  
Control # 27

YEARS IN CURRENT JOB ? 1.5 yr

DESCRIBE JOB

primarily responsible for production and fielding of system HW and SW;  
everything related to that, including reliability improvements to HW and  
SW

-----  
Control # 28

YEARS IN CURRENT JOB ? 6 mo

DESCRIBE JOB

technical lead on all computer resources , HW and SW; documentation  
review; design review; anything in the computer on this effort, he is  
responsible for; more toward a management perspective, less technician  
(interviewee is in an engineering position)

-----  
Control # 30

YEARS IN CURRENT JOB ? 4 yr

DESCRIBE JOB

computer resources systems engineer; responsible for overseeing policy  
and procedures for SW acquisition on program--technical perspective;  
compliance with ASD policies

-----  
Control # 31

YEARS IN CURRENT JOB ? 4 yr

DESCRIBE JOB

project management activities related to the program (cost, schedule);  
active participant in CRWG (sometimes chair); chair of ICWG

-----  
Control # 9B

YEARS IN CURRENT JOB ? 12 yr (b)

DESCRIBE JOB

avionics subsystem engineer

-----  
Control # 12B

YEARS IN CURRENT JOB ? 6 yr (b)

DESCRIBE JOB

computer resources systems engineer; policy type job--develop, implement  
policies; technical help whenever needed; computer resources focal point

-----  
██

QUESTION 22

Control # 4

TYPES OF INFORMATION TO HELP WITH JOB ?

weekly software review minutes, contractor plant visits

-----

Control # 5

TYPES OF INFORMATION TO HELP WITH JOB ?

software development account document; software status report; QA reports; V&V report from WR-ALC item managers (monthly); inputs from SM-ALC (system manager); meeting and review minutes from CRWG; design reviews, TIMs; SW TIMs

-----

Control # 6

TYPES OF INFORMATION TO HELP WITH JOB ?

OO-ALC doing IV&V (reports); CPR->WBS structured adequately enough (cost data); quarterly PMRs and monthly program progress reports; telecons; design reviews and tech meetings between prime and sub contractors

-----

Control # 8

TYPES OF INFORMATION TO HELP WITH JOB ?

SDP; SW status report; SW development accounting document (database); (will get) source code; SW productivity consortium handbook

-----

Control # 9A

TYPES OF INFORMATION TO HELP WITH JOB ?

A: specifications and documents

-----

Control # 10

TYPES OF INFORMATION TO HELP WITH JOB ?

(indicators are primary source) feedback from test data

-----

Control # 12A

TYPES OF INFORMATION TO HELP WITH JOB ?

A: office automation, MIS system in-house; other mechanisms besides TPMs--design reviews (technical), PMR (management type), carry forward issues at SDCCR into additional reviews

-----

Control # 13

TYPES OF INFORMATION TO HELP WITH JOB ?

formal: documentation from contractor; informal: list of all problem reports; source code listings (on demand)

-----

Control # 14

TYPES OF INFORMATION TO HELP WITH JOB ?

integrated master plan, integrated master schedule; CDRLS -> SW documentation specification, SW test plan, SW design documentation; regular program reviews (PDR,CDR)

-----

Control # 15

TYPES OF INFORMATION TO HELP WITH JOB ?

information from other product teams; a dedicated embedded software person; information management key factor; dissemination is key; tools

make it more useful (commonality of automated information management systems)

-----  
Control # 16

TYPES OF INFORMATION TO HELP WITH JOB ?

verbal updates from engineers on program status (informal contact)

-----  
Control # 17

TYPES OF INFORMATION TO HELP WITH JOB ?

verbal contact with contractor counterpart (weekly)

-----  
Control # 18

TYPES OF INFORMATION TO HELP WITH JOB ?

verbal information from contractor personnel; conversations with lower levels of contractor organization to get "bottom-line" opinions

-----  
Control # 19

TYPES OF INFORMATION TO HELP WITH JOB ?

verbal with contractor; design reviews and monthly assessments; contractor plant visits; CRDL items; documentation reviews (SRS, SDS)

-----  
Control # 20

TYPES OF INFORMATION TO HELP WITH JOB ?

weekly meetings with contractor, e-mail minutes to meetings; e-mail primary means of communications; TIMs every six weeks; technical performance measures

-----  
Control # 21

TYPES OF INFORMATION TO HELP WITH JOB ?

current news to see what is happening in the corporate environment for computer companies; IEEE publications; AFSC newsletters

-----  
Control # 23

TYPES OF INFORMATION TO HELP WITH JOB ?

status reports; assessments; plant visits; \*\*communication is most important aspect of doing your job, no metric stands alone, communications brings it all together, finds out details on metrics

-----  
Control # 24

TYPES OF INFORMATION TO HELP WITH JOB ?

STSC tech reports (458-8045, SW management guide MACOIO/ST-E); MCCR handbook; data analysis center for software (DACS) report; monthly status report (not really useful, better if it had indicators); verbal status reports

-----  
Control # 25

TYPES OF INFORMATION TO HELP WITH JOB ?

contractor documentation; verbal feedback from program office, contractor, system operating location; design plans

-----  
Control # 26

TYPES OF INFORMATION TO HELP WITH JOB ?

integrated product team meetings (contractor and SPO) discuss status, slips, problems, new developments; meeting minutes

-----

Control # 27

TYPES OF INFORMATION TO HELP WITH JOB ?

large portion of evaluation is done by test agencies, DT&E and OT&E--use their inputs, ALC inputs, verbal inputs, field visits; key meeting is the operational test certification meeting

-----

Control # 28

TYPES OF INFORMATION TO HELP WITH JOB ?

contractor data submittals, design reviews, face-to-face meetings

-----

Control # 30

TYPES OF INFORMATION TO HELP WITH JOB ?

tracking of versions that are planned for lab testing and future builds; contractor sends (faxes) in projections of future builds

-----

Control # 31

TYPES OF INFORMATION TO HELP WITH JOB ?

proactive relationship with contractor; close working relationship with contractor counterpart; contractor visits (1 week in 6); attend PMRs and TIMs; share information with contractor

-----

Control # 9B

TYPES OF INFORMATION TO HELP WITH JOB ?

design analysis reports; test reports; specs; schedules; FCAs; test requirements matrix

-----

Control # 12B

TYPES OF INFORMATION TO HELP WITH JOB ?

development team leaders send information; internal MIS EMail and bulletin boards for ACSN, ECP, CDRLs and TPMs

-----

QUESTION 23

-----

Control # 4

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

requirements traceability and testing, automated tools or anything else that could help with these functions

-----

Control # 5

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

tradeoff was done to accept SW development account document (SDAD) instead of "formal" documentation means that new people do not have the documentation to bring them up to speed easily, some people feel uncomfortable with this approach; SDAD requires special terminal to access (interviewee was not uncomfortable with approach)

-----

Control # 6

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?



need training in Ada, object-oriented design, and SW cost estimation;  
need "workbook" for process involved in getting "good" metrics, lessons  
learned, and what to do with data

-----

Control # 8  
HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?  
almost "too much" information

-----

Control # 9A  
HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?  
(none)

-----

Control # 10  
HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?  
none, really; fairly good contractor to work with, good working  
relationship with POCs

-----

Control # 12A  
HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?  
electronic network not fully in place; secure system ok, but need  
multilevel (unclassified too) system also--integrated master plan needed  
as part of system; need face-to-face time with contractor because of  
team style; integrated master schedule is not contracted for

-----

Control # 13  
HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?  
paper product of source listing--not really, can get this informally  
from contractor; formal report of trouble reports and their closure,  
formal complexity report would be nice periodically

-----

Control # 14  
HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?  
trying to get "same set" of software tools; need more tools to allow for  
automated analysis; establish historical database of past performance to  
be used for program requirements, comparison of actual delivered vs.  
estimates; "black box" of RCA software models unsettling; unable to  
analyze code across different languages

-----

Control # 15  
HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?  
need better dissemination and tools; issues

-----

Control # 16  
HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?  
top-level design documents (not too much detail); documentation for  
maintainability to check for completeness and user satisfaction (usable  
format); too much documentation on contract now

-----

Control # 17  
HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?  
more information from test organization on performance of SW  
(anomalies); something from engineer on what they do with SW; more

information from  
quality group about quality review results

---

Control # 18

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

important to get indicators on contract; some way to specify indicators

---

Control # 19

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

wanted correct indicators, if had them, job would be OK

---

Control # 20

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

current information is adequate, with CS\*\*2, TPMs, e-mail, SDIP producing all the information needed

---

Control # 21

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

no further information needed right now

---

Control # 23

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

specification system (CSPEC) needs to be changed to adapt to track things required for metrics

---

Control # 24

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

indicator data in monthly status report; need standard indicators and method for getting in contracts -> monthly status report, special DID for indicators should be mandatory; courses to interpret and apply indicators; management recognition of indicators

---

Control # 25

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

opposite is true--suffer from "information overload"; need easier data to process

---

Control # 26

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

communication biggest issue, getting timely information out to everyone is a factor; need secure network

---

Control # 27

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

for new efforts, it is difficult to get hands around complexity--lines of code doesn't give it to you--our cost models some times come in at 10% of what the contractor estimates

---

Control # 28

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

problem becomes getting too much information, and sifting out the really important items becomes difficult

-----  
Control # 30

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

need information on projected deliveries of documentation and completeness of documentation  
-----

Control # 31

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

would "love to have seen" integrated schedules; PERT and CPM techniques applied across multiple developments in the program (projects seemed managed in an isolated fashion)  
-----

Control # 9B

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

(none)  
-----

Control # 12B

HOW DOES THIS INFORMATION DIFFER FROM WHAT YOU NEED ?

integrated master plan needed as part of electronic system; need face-to-face time with contractor due to team style; master schedule is not contracted for  
-----

QUESTION 24  
-----

Control # 4

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?

pass on weekly software review meeting minutes; verbal on software testing results and status; identification of software problem areas in document  
status  
-----

Control # 5

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?

SW team report (with ALCs and ASD) reports on all activities during month and assesses data deliveries, is sent to program manager; as necessary briefings  
-----

Control # 6

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?

quarterly program reviews (4-letter); monthly contractor performance assessment reporting (system); keep program manager informed  
-----

Control # 8

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?

reports; prepare briefing charts for ALC; verbal status updates  
-----

Control # 9A

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?

verbal, memos  
-----

Control # 10

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
provide monthly status briefings to program manager; verbal reports to  
immediate manager

-----

Control # 12A

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
very little formal products--most products come out of contractor;  
program status reviews; front-to-back; action item status  
management technical information system provides upper level visibility  
automatically; trip reports, meeting minutes

-----

Control # 13

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
informal verbal updates, memos

-----

Control # 14

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
weekly "top 10" issues on a watch list, tracked closely; trip reports

-----

Control # 15

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
trip reports; IPR issues-> tells where comes from (SOW, IMP, IMS);  
verbal telecons; status briefings

-----

Control # 16

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
provide knowledge of priority one SPRs to higher-level management (could  
be formal or informal); provided briefing slides; verbal updates and  
memos;  
tracking of open/closed SPRs

-----

Control # 17

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
indicator data (no response from higher management yet); trip reports

-----

Control # 18

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
trip reports; SW schedule and test process briefings

-----

Control # 19

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
briefings; informal trip reports; verbal updates; overall assessments  
when required

-----

Control # 20

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
verbal updates; there is an executive metric briefed; one metric shows  
SW ready to "go into the can"; there is a lot of history with the  
program, as development could be considered third time around, due to  
two previous prototypes

-----

Control # 21

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
trip reports; briefings; verbal updates

-----  
Control # 23

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
brief in staff meeting (weekly); weapon system review (EMD team) (monthly)

-----  
Control # 24

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
bi-weekly team meetings, briefings with engineering and management;  
status reports; charts/letters; address user concerns

-----  
Control # 25

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
document reviews/acceptances; respond to verbal inquiries

-----  
Control # 26

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
put out newsletter to let everybody know what's going on with project;  
trip reports to higher management; staff meetings

-----  
Control # 27

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
memos to tell supervisor what is going on, also memos to team members;  
work on requirements document from ACC

-----  
Control # 28

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
briefing charts; weekly meeting with SPO; "white papers"; bi-monthly  
PMRs, including "government only" session

-----  
Control # 30

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
trip reports; approval status of all SW documentation; status of  
airworthiness reviews of SW that has been informally tested by the  
contractor previously; generate a status "slide" that displays the  
current status against the incremental build plan

-----  
Control # 31

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
typical program management reporting: trip reports memos E-Mails--primary  
emphasis on briefings

-----  
Control # 9B

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
verbal, memos

-----  
Control # 12B

WHAT TYPES OF PRODUCTS DO YOU PRODUCE FOR HIGHER MANAGEMENT ?  
briefing at program status review; on-spot briefings; informal EMail  
updates

## QUESTION 25

-----  
Control # 4

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

it is more important for the contractor to use the indicators than for the government to have them delivered, the government can't do much of anything for the contractor to help them

-----  
Control # 5

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

indicators need to be geared toward all phases of the contract, and should be planned accordingly

-----  
Control # 6

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

\*what to do with data; how to book/workshop on metrics

-----  
Control # 8

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

-----  
Control # 9A

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

A: specification coverage in testing

-----  
Control # 10

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

do not rely on indicators heavily (exclusively), still need to fix things; documents/test procedures

-----  
Control # 12A

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

A: tailored to meet phase and need of programs; knowledgeable people to interpret indicators; thoughtfully pick and make sure satisfy needs; you get what you measure; make sure indicators are balanced--evenly spaced (phase dependency); work package tracking done in SDP; give to CSSC tracks IMS; standard set does not make sense

-----  
Control # 13

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

-----  
Control # 14

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

\*\* more information on how to use indicators-- training; automated tools/listing of metrics; put in electronic media that can be shared and broad stimuli to make changes

-----  
Control # 15

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

emphasis on use of indicators for specific purposes and outcomes, not for the sake of having them; the results of using metrics

Control # 16

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

design stability really indication of requirements; not driven by calendar as much as manager, don't want cost too high, but want to do the job right; maintainability

-----

Control # 17

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

manpower both expertise and staffing levels (right person for the right job), and requirements; empathy for SW in general; general lack of knowledge of indicators; better access to classes, signed up for 3, yet could not get into any

-----

Control # 18

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

-----

Control # 19

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

should be training on how to use indicators

-----

Control # 20

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

accuracy of data is important, on-line metrics gathering really helps this effort

-----

Control # 21

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

-----

Control # 23

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

planning parameters updating

-----

Control # 24

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

stress indicators should be part of every contract (at a minimum)

-----

Control # 25

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

indicators not stressed enough in education process; no bridge between the folks that do it and the folks that use it (indicators); maybe some kind of checklist is needed; things done too much in isolation

-----

Control # 26

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

process compliance indicator--trouble defining it adequately (what is significant, what is not); some indicators need more definition and guidance on how to use them

-----

Control # 27

ANYTHING ELSE YOU FEEL IS IMPORTANT ?

interviewee's area of concern--thought process that goes into smart coding of software--no plans for looking at what constraints are and

getting code into constraints--there appears to be a short term versus long term look into the limitations, leading to expensive recoding efforts later on

-----

Control # 28  
ANYTHING ELSE YOU FEEL IS IMPORTANT ?

-----

Control # 30  
ANYTHING ELSE YOU FEEL IS IMPORTANT ?  
need to stress the concept of resource model vs cost model, and look at all resources in combination--cost, manpower etc, are all resources

-----

Control # 31  
ANYTHING ELSE YOU FEEL IS IMPORTANT ?  
\*metrics are not an end unto themselves, but they are one of the "tools" in your toolbox; need to get the right people to work for you

-----

Control # 9B  
ANYTHING ELSE YOU FEEL IS IMPORTANT ?  
measuring of proposals vice reality

-----

Control # 12B  
ANYTHING ELSE YOU FEEL IS IMPORTANT ?  
purpose and driving need--tailoring to needs and phases-- not a mandate, but suggest for tailoring; quality measurement of metrics ; work package tracking percentage completion criteria for each atomic action (tracks to technical schedules); knowledgeable personnel



## Appendix D: Survey Responses

### Appendix D.2: Without Indicators Survey Responses

This database has been sorted by question number. Not all interviewees were asked all questions due to survey revisions (response normally left blank). In addition, not all interviewees responded to every question asked. The first six questions were programmatic data and have been grouped at the beginning of the report.

#### Control # 1

CONTR AWARD 88-trace a/b  
CONTRACT END

CONTRACT TYPE  
COST PERF REPORT

PROG PHASE EMD part a, b mods  
EST SIZE

#### Control # 2a/b

CONTR AWARD  
CONTRACT END 94 part 2 (est)

CONTRACT TYPE  
COST PERF REPORT

PROG PHASE production  
EST SIZE

#### Control # 3

CONTR AWARD 84  
CONTRACT END 94 pt2,pt1 done

CONTRACT TYPE  
COST PERF REPORT

PROG PHASE Production  
EST SIZE

#### Control # 7

CONTR AWARD 93- pre-RFP now  
CONTRACT END approx 96

CONTRACT TYPE FFP (IF?)  
COST PERF REPORT

PROG PHASE EMD  
EST SIZE

Control # 11

CONTR AWARD 88 OFP tied prd  
CONTRACT END

CONTRACT TYPE FFP +AF  
COST PERF REPORT

PROG PHASE EMD & Prod, 3010 \$\$  
EST SIZE

Control # 22

CONTR AWARD late 88  
CONTRACT END PMRT in 92

CONTRACT TYPE FFP  
COST PERF REPORT? NO

PROG PHASE EMD/production run 2 units  
EST SIZE 70 KLOC, est was 100

Control # 29

CONTR AWARD 87  
CONTRACT END 92

CONTRACT TYPE CP  
COST PERF REPORT? YES

PROG PHASE 1/2 done, 1/2 undone  
EST SIZE 100 KLOC

---

Control # 1

WHAT ARE THE PRIME REASONS WHY THERE ARE NO INDICATORS ON  
THE PROGRAM?

Cost, limited budget (even CACRL items being cut)  
Not big push when contract was awarded

---

Control # 2a/b

WHAT ARE THE PRIME REASONS WHY THERE ARE NO INDICATORS ON  
THE PROGRAM?

SW baselined during FSD, now mostly corrections and  
enhancement; SW matured enough

---

Control # 3

WHAT ARE THE PRIME REASONS WHY THERE ARE NO INDICATORS ON THE PROGRAM?

was not much emphasis on indicators at time of contract award; cost

---

Control # 7

WHAT ARE THE PRIME REASONS WHY THERE ARE NO INDICATORS ON THE PROGRAM?

Plan on using in future for risk, schedule, cost measurement

---

Control # 11

WHAT ARE THE PRIME REASONS WHY THERE ARE NO INDICATORS ON THE PROGRAM? subsystem managers track schedule and funding; tracking being done at a lower level

---

Control # 22

WHAT ARE THE PRIME REASONS WHY THERE ARE NO INDICATORS ON THE PROGRAM?

near end of program, not much value added at this point; tried indicators early on because management wanted ; tried LOC, difficult as estimated "floated" (ambiguous definitions \*\* [replicated code, when was code "completed," etc] led to ambiguous statements by contractor), easier to get a handle on software modules (CSUs, CSCs)

---

Control # 29 WHAT ARE THE PRIME REASONS WHY THERE ARE NO INDICATORS ON THE PROGRAM?

program is an upgrade of an existing system that was done in the early 80's, and only way program would have indicators is if the original program had them--interviewee believed that program did not

---

Control # 1

HOW DO YOU PREDICT SCHEDULE, COST, PRODUCTIVITY, AND PRODUCT QUALITY?

informal: close contact with subcontractor weekly phone/TDY (visit contractor/subs monthly). Had problem when prime stopped flow of info

formal: brief same info at quarterly PMRs -> deficiencies open/closed, schedule, personnel, CDRL schedule

---

Control # 2a/b

HOW DO YOU PREDICT SCHEDULE, COST, PRODUCTIVITY, AND PRODUCT QUALITY?

contractor database online access to SDRs; SPO PM gives info; day to day conversations; baselined contractor has intend benchmarking, then goes to simulator; evaluate SW parameters with Safety and Interoperability Branches; flight safety office, earlier part of system more tested

---

Control # 3

HOW DO YOU PREDICT SCHEDULE, COST, PRODUCTIVITY, AND PRODUCT QUALITY?

use historical program data; informal data inputs from contractor; some info from SQA on deficiency reports; access to contractor's database on prime equipment SDRs tracked against problem and document (but may not be accurate->behind reality); contractor monitors product quality; monitor testing

---

Control # 7

HOW DO YOU PREDICT SCHEDULE, COST, PRODUCTIVITY, AND PRODUCT QUALITY?

2167A milestones--look for completeness of documents for phase completion

---

Control # 11

HOW DO YOU PREDICT SCHEDULE, COST, PRODUCTIVITY, AND PRODUCT QUALITY?

interact with funding arena; scheduling; work with subsystem managers (Capts interviewed later)

---

Control # 22

HOW DO YOU PREDICT SCHEDULE, COST, PRODUCTIVITY, AND PRODUCT QUALITY?

FFP set progress payments at a small number of points -> at each point evaluate whether or not milestone has been completed; learned from experiences, costs underestimated, at CDR get manpower estimate, but did not meet that estimate, slipped out schedule -> began tracking how using manpower began plant visits, used DCAS; correlation of manpower to how we are doing, schedule slippage to manpower estimates; deliberate contractor undermanning to cover other efforts

---

Control # 29

HOW DO YOU PREDICT SCHEDULE, COST, PRODUCTIVITY, AND PRODUCT QUALITY?

CSSR provides cost and schedule information (monthly (?)); TIMs give status and information, bi-monthly; functional readiness review demonstrates product; benchmark tests (written by interviewee) would have identified quality of software, provided a baseline, and would have been foundation for FQT

---

Control # 1

DO YOU ENVISION SOFTWARE INDICATORS BEING USED IN THE FUTURE ON THE PROGRAM? no

contract too close to completion. Money (may be biggest factor), contractor relationship bad

---

Control # 2a/b

DO YOU ENVISION SOFTWARE INDICATORS BEING USED IN THE FUTURE ON THE PROGRAM? NO

physical size about maxed, no more room for major SW effort; asks for more capability that requires it probably won't happen; really looking at another effort (follow-on

to original program); software is stable

---

Control # 3

DO YOU ENVISION SOFTWARE INDICATORS BEING USED IN THE FUTURE  
ON THE PROGRAM? NO  
very little SW development at this stage in program

---

Control # 7

DO YOU ENVISION SOFTWARE INDICATORS BEING USED IN THE FUTURE  
ON THE PROGRAM? yes  
see q7

---

Control # 11

DO YOU ENVISION SOFTWARE INDICATORS BEING USED IN THE FUTURE  
ON THE PROGRAM?

---

Control # 22

DO YOU ENVISION SOFTWARE INDICATORS BEING USED IN THE FUTURE  
ON THE PROGRAM? yes  
A manpower indicator would be most useful, in retrospect;  
useful to track slips

---

Control # 29

DO YOU ENVISION SOFTWARE INDICATORS BEING USED IN THE FUTURE  
ON THE PROGRAM? NO  
contract terminated--cost overruns

---

Control # 1

YEARS INVOLVED WITH SOFTWARE 3                      TRAINING FOR JOB

CRAC basic and advanced; ENASC in-house course; seminar on 2167/2168; Keesler ATC short course SW quality assessment, structured software development; BSCE

---

Control # 2a/b

YEARS INVOLVED WITH SOFTWARE a-7 b-3                      TRAINING FOR JOB

A: BSCS, AFIT short course SW engineering B: BSEE A/B: OJT biggest factor

---

Control # 3

YEARS INVOLVED WITH SOFTWARE 10                      TRAINING FOR JOB

CSOC; BSCS; sys 212; MCCR management; OJT

---

Control # 7

YEARS INVOLVED WITH SOFTWARE 17                      TRAINING FOR JOB

BS sys eng; courses in acq of SW; programming language courses; cost management courses

---

Control # 11

YEARS INVOLVED WITH SOFTWARE 1                      TRAINING FOR JOB

basis of knowledge is operational, flown missions in F-4 and F-111; user interaction knowledge; OFP knowledge; control systems major in college; (more trouble quote moved to Q21)

Control # 22

YEARS INVOLVED WITH SOFTWARE 6 TRAINING FOR JOB

BSEE; OJT started in trainers (SW); 2167 course; CRAC,  
advanced CRAC; tested out of sys 100,200

---

Control # 29

YEARS INVOLVED WITH SOFTWARE 30yr (civ) TRAINING FOR JOB

systems engineering background; OJT; civilian experience;  
sys100

---

Control # 1

TIME IN CURRENT JOB? 3 DESCRIBE JOB

POC for software for two sub systems; review data items;  
inform the team leader on status of SW testing; keep up to  
date on status; witnessing test of SW deliveries; SW QA,  
review FCA/PCA documentation; I/F with liaison; review  
CDRLs; monitor deficiencies to see if they are out of scope

---

Control # 2a/b

TIME IN CURRENT JOB? a-3 b-3 DESCRIBE JOB

SW documentation review; issue management; audit support;  
safety studies; I/F with other EN branches (i.e. radar);  
work other support for EN; systems interface (with airframe,  
etc); ICWG participant; review B5, C5; proposal review;  
component study

---

Control # 3

TIME IN CURRENT JOB? 2.5 yrs DESCRIBE JOB

evaluate SW inputs to proposals (ECPs); interface with  
contractor; planning and scheduling for future SW upgrades  
(flight testing, test labs); coordinate SW schedule with  
aircraft managers; develop plan for IWSM organic SW; ensure  
commonality of SW; daily contact with contractor SW  
personnel

---



Control # 7

TIME IN CURRENT JOB? 1.5 yrs DESCRIBE JOB

responsible for all performance issues; and defining what those requirements are, documenting, contracting, testing, and delivering them; influenced indirectly by cost and schedule; performance issues

---

Control # 11

TIME IN CURRENT JOB? 1 yr DESCRIBE JOB

capability review; overall integration perspective; monitor customer needs

---

Control # 22

TIME IN CURRENT JOB? 3 yr DESCRIBE JOB

lead training systems integration engineer; help define requirements; develop test program to verify requirements; review contractor's progress; de facto assistant PM--i.e. track schedule, answer questions; default test monitoring person

---

Control # 29

TIME IN CURRENT JOB? 6 yr DESCRIBE JOB

chief engineer for program; all engineering responsibilities; technical briefings; interface with contractor counterpart; overall liaison with program manager, program control, contracting, and contractor; documentation approval

---

Control # 1

WHAT TYPES OF INFORMATION DO YOU GET TO DO YOUR JOB?

information at PMRs; CDRL submittals; minutes of meetings; telephone contact on status of deficiencies, schedule

Control # 2a/b

WHAT TYPES OF INFORMATION DO YOU GET TO DO YOUR JOB?

when they had IV&V, got reports from IV&V on documentation.

---

Control # 3

WHAT TYPES OF INFORMATION DO YOU GET TO DO YOUR JOB?

schedule from contractor; phone conversations;  
problem reports; MIPs and SRs; ECPs and CCPs

---

Control # 7

WHAT TYPES OF INFORMATION DO YOU GET TO DO YOUR JOB?

indicator data; audit reports from DCAS-> compare  
for accuracy to contractor reporting system;  
validate requirements; monitor large system  
utilization, see how much and who is using systems

---

Control # 11

WHAT TYPES OF INFORMATION DO YOU GET TO DO YOUR JOB?

---

Control # 22

WHAT TYPES OF INFORMATION DO YOU GET TO DO YOUR JOB?

"indicators"; information at design reviews;  
periodic scheduled inputs (monthly)->not formal  
-but information on ;plant visits;  
information from test results; tests show defects,  
act like DT&E tests; specific estimates on  
specific tasks

---

Control # 29

WHAT TYPES OF INFORMATION DO YOU GET TO DO YOUR JOB?

TIMs (formal and informal); daily conversations  
with contractor counterpart; PMRs

---

Control # 1

HOW DOES THIS INFORMATION YOU GET DIFFER FROM WHAT YOU  
NEED?

throughput (in beginning of program); status of deficiencies  
and schedule (especially around flight testing time);  
reporting more frequently than PMR on requirements  
completion, design progress, coding (in beginning), schedule

---

Control # 2a/b

HOW DOES THIS INFORMATION YOU GET DIFFER FROM WHAT YOU  
NEED?

not information, but manpower can be a big issue; additional  
documentation review help

---

Control # 3

HOW DOES THIS INFORMATION YOU GET DIFFER FROM WHAT YOU  
NEED?

none; can get what is needed from contractor

---

Control # 7

HOW DOES THIS INFORMATION YOU GET DIFFER FROM WHAT YOU  
NEED?

get whole system automated; reports in on disk, can get  
farther without paper; \*\* more formalized DIDs and CDRLs \*\*  
formatted for automated data delivery (indicator  
collection); tailorable DID for indicators

---

Control # 11

HOW DOES THIS INFORMATION YOU GET DIFFER FROM WHAT YOU NEED?

cost data->insight into subsystems progress; user guides big deal to help understand; need budget flexibility; out of scope problems that can be solved for less than \$250K covered under MOA.

---

Control # 22

HOW DOES THIS INFORMATION YOU GET DIFFER FROM WHAT YOU NEED?

through time, narrowed down what was needed, -> contractor ; lacking in clout to force contractor to do work contractor in Ch11

---

Control # 29

HOW DOES THIS INFORMATION YOU GET DIFFER FROM WHAT YOU NEED?

would like to get more accurate, better quality information, especially at PMRs

---

Control # 1

WHAT TYPES OF INFORMATION PRODUCTS DO YOU PRODUCE?

oral reports; MFR if required

---

Control # 2a/b

WHAT TYPES OF INFORMATION PRODUCTS DO YOU PRODUCE?

provide status summary during FCA/PCA; documentation review summaries (not much going on right now)

---

Control # 3

WHAT TYPES OF INFORMATION PRODUCTS DO YOU PRODUCE?

minutes of meetings (CRWG, ICWG); status briefing to SPO director; inputs to team meetings (various associated systems. . . ); was providing status updates

---

Control # 7

WHAT TYPES OF INFORMATION PRODUCTS DO YOU PRODUCE?

all documents tracked-> suspense tracking report for ECPs, CCPs, data submissions; weekly have formal report produced

---

Control # 11

WHAT TYPES OF INFORMATION PRODUCTS DO YOU PRODUCE?

\*road show briefing, prior to OFP release, critical to safety of pilot

---

Control # 22

WHAT TYPES OF INFORMATION PRODUCTS DO YOU PRODUCE?  
schedule estimates; briefing reviews given to management (monthly or quarterly); \*simplistic measures of progress

---

Control # 29

WHAT TYPES OF INFORMATION PRODUCTS DO YOU PRODUCE?

briefings to program manager and to industry; papers for technical journals

---

Control # 1

ANYTHING ELSE YOU FEEL IS IMPORTANT TO THE USE OF INDICATORS?

development phase, contractor willingness to use indicators; firm fixed price contracts (more need if not FFP); cost; point in development

---

Control # 2a/b

ANYTHING ELSE YOU FEEL IS IMPORTANT TO THE USE OF INDICATORS?

sizing tied back to development, now requirements have been overshadowed and don't have 30% ; production phase spec variance, traceback into design

---

Control # 3

ANYTHING ELSE YOU FEEL IS IMPORTANT TO THE USE OF INDICATORS?

looked closer at problems when testing problem with definitions of open and closed by MMC so wasn't useful; \*\* if generating themselves, how much labor is involved?; benefits?; SPRs generated during testing

---

Control # 7

ANYTHING ELSE YOU FEEL IS IMPORTANT TO THE USE OF INDICATORS?

tailorability of indicators to size of program; level of appropriateness, guidance provided

---

Control # 11

ANYTHING ELSE YOU FEEL IS IMPORTANT TO THE USE OF INDICATORS? from q 13: more trouble with software not paying attention to users needs->network of participants->brain management

\*be careful of losing flexibility, difference between what user wants and direction to contractor, "out of scope" changes -> MOA to fix minor changes without ECP proven; repository -> have to understand how to use them (indicators)!! -> active management -> go to customer, need face to face!!; as manpower decreases -> becomes critical!!; question of effectiveness of database; database needs to be managed; database manager has to have customer interface to get information; lessons learned

---

Control # 22

ANYTHING ELSE YOU FEEL IS IMPORTANT TO THE USE OF  
INDICATORS?

---

Control # 29

ANYTHING ELSE YOU FEEL IS IMPORTANT TO THE USE OF  
INDICATORS?

maybe indicators belong in 2167 or 2168--this would put the  
indicators where they would be used--an appendix of  
indicators

### Appendix D.3 ASC/ENASC Survey Responses

**1. Who originated the database effort?**

internally generated--EN(CR) advocated effort, but not much activity; ACC had responsibility, but weren't too involved (wanted cost data) problems with the WBS, ESD solution too expensive EN does ICA, but wanted to limit their effort.

**2. When was it originated?**

2 years ago

**3. Why was it originated?**

3 reasons:

a. independent assessments (comparisons) of programs(size, schedule) based on requirements, bounce against budget & schedule & time

b. surveys of software and processors, tool sets-- general information about ASC programs

c. 1750A architecture validation

**4. What are the primary objectives of the database?**

see question 3

**5. Is there any formal documentation identifying the need for the database, validating it as a requirement, or providing tasking for it?  
YES / NO If YES, can we get a copy?**

No. Done within available resources--functional need statement

**6. Who do you foresee as the "customer" of the database effort?**

Program offices that need support--customer direct access

**7. What do you believe the customer needs from the database?**

a. software size

b. size growth

c. schedule/schedule estimates

d. memory/total and actual



8. What do you expect to provide the customer?

- a. trend analysis->comparison of different programs
- b. size estimates (realism)
- c. schedule estimates

9. How do you expect to provide it?

- a. inputs to briefings
- b. written reports
- c. working directly with individuals from program offices who generates own data

**FOLLOW-UP ACTIONS THAT NEED TO BE ACCOMPLISHED:**

Öáç

âáì Have formal documentation for database

---

---

---

---

---

---

---

---

---

---

## ENASC SURVEY

Please review the following areas of software development and rate them based on the scale below in terms of how significant they are to the database:

1	2	3	4	5
Very				Very
Significant	Significant	Neutral	Unsignificant	Unsignificant

- \_\_\_ Schedule/Project Tracking
- \_\_\_ Reliability
- \_\_\_ Cost Projection/Estimation/Tracking
- \_\_\_ Software Performance
- \_\_\_ Quality
- \_\_\_ Manpower
- \_\_\_ Testing
- \_\_\_ Requirements Traceability/Stability
- \_\_\_ Maintainability
- \_\_\_ Documentation

## Appendix D.4 Database Customer Survey Responses

When did you use the ENASC database?

1. 2 months ago
- 2.
- 3.
- 4.

What stage of the program were you in?

1. Pre-RPF, pre-contractual
2. Platform Competition
- 3.
4. Concept exploration

What were your specific objectives/ needs in using the database?

1. Cost estimates/schedules; use of REVIC, SEER, System 4
2. Comparison as with respect to other programs; rough order magnitude idea--sizing aspect
3. ENASC ran REVIC/SEER, cost estimation models for user. He was looking for schedule months, costs for a software upgrade--estimates
4. Preliminary WAG of software cost/schedule

Were these objectives/needs satisfied? Why/Why not?

1. Yes
2. Yes
3. Yes, but now the user has REVIC and does not need their assistance anymore for that.
4. Yes, was able to run REVIC

Is there anything you would like to see added to/ changed in the database?

1. Generally satisfied--would like to see more correlation between models
2. Day to day utility would require high level of inputs--there are cost problems and walls between the programs that discourage this (manpower issues, contractor performance)--you could get overkill with the database
3. Not right now, expert help for other people useful
4. Databases are not useful. Technology moves rapidly, difficult to do a search-- level of complexity-of systems not equivalent (did not use database for sizes). More value from having several models and knowing how to use them. We do not contract for enough data to populate a database.

Would you use the database again? Why/Why not?

1. Yes, ENASC are the experts
2. Yes, they are his home office, wants to get them in the loop, learn lessons from other programs.
- 3.
4. Would use again for cost model, database not of use--except maybe to calibrate cost models(?)

## Appendix E Results of Analysis of Interview Questions

### WITHOUT INDICATORS

#### Question 15--Transform

##### Engineers

review documentation	17%
keep higher management informed	4%
testing actions	9%
software quality assurance	4%
interface with others	17%
identify problems	4%
technical assistance	9%
safety	9%
requirements	9%
software performance	4%
track schedule	4%
briefings	4%
work with contractor	4%

##### Manager

ECP evaluation	33%
work with contractor	33%
plan/schedule	33%

#### Question 8--Transform

telecons	14%
plant visits	9%
reviews	18%
online access to database	14%
information from PM	9%
input from other teams	18%
historical data	5%
data from contractor	5%
CSSR	5%
TIMs	5%

#### Question 9--Outputs

verbal reports	8%
memos	8%
activity status reports	33%
meeting minutes	8%
briefings	25%

schedule estimates	8%
technical papers	8%

Question 7--Voice of the Customer

cost	22%
no emphasis	22%
software mature	22%
end of program	11%
tried but failed (ambiguous definition)	11%
continuation of older contract	11%

Question 11--Voice of the Customer

too close to completion	14%
cost	14%
bad contractor relationship	14%
additional software development unlikely	43%
contract terminated	14%

Question 18--Voice of the Customer

throughput	20%
schedule	20%
SPRs	20%
electronic data delivery	20%
better, more accurate information at PMRs	20%

Question 20--Voice of the Customer

contractor willingness	13%
cost of indicators	25%
phase of development	13%
how to define open/closed SPRs	13%
tailoring indicator to size of program	13%
guidance on indicators	25%

Question 17--Inputs

reports	17%
meeting minutes	6%
briefing at review	17%
CDRL items	11%
plant visits	6%
telecons	17%
TIMs	6%
test agency inputs	6%
integrated master schedule	17%

### WITH INDICATORS

#### Question 10--Inputs

gets reports	35%
gets meeting minutes	7%
gets database copy	7%
attends briefings at reviews	10%
receive CDRL items	2%
contractor forwards to SPO	10%
telecons	12%
quarterly discussions at meetings	2%
database access	7%
access to primitives	7%
joint (contractor SPO) effort	2%

#### Question 22--Inputs

reports	14%
meeting minutes	4%
database copy	1%
briefing at review	14%
CDRL items	13%
plant visits	6%
ALC inputs	3%
joint effort	1%
telecons	8%
TIMs	6%
source code	3%
test agency inputs	4%
face-to-face	1%
publications	6%
MIS system	3%
E-mail	3%
integrated master plan	1%
integrated master schedule	1%
other teams	7%

#### Question 16--Voice of the Customer

##### yes

change to better represent desired information	78%
reflect changes in new requirements	11%
reflect changes in support activities	11%

##### no

late stage of program	50%
contractor refuses	10%
program going well--no need	40%

#### Question 12A--Voice of the Customer

##### yes

good insight	30%
watch spec requirements	9%
cross-check proposal	4%
highlight deviations from plan	9%
provide consensus as to phase/ milestone completion	4%
control budget/schedule	4%
provided current status	17%
foresee problems	9%
structured approach to getting information	4%
leverage against contractor	4%
get management excited	4%

##### no

more for contractor than for SPO	33%
data too late for action	17%
could get data from other sources	17%
for higher-level use	17%
indicators provide information for use by management, not by engineering	17%

#### Question 15--Voice of the Customer

contractor refuses to provide	7%
delete non-valuable indicators	27%
change indicator to better represent desired information	60%
recommendation on independent review team	7%

#### Question 7--Voice of the Customer

provide status/monitor progress	28%
verify contractor applying resources correctly	2%
insight	6%
look for trends	6%
compare against plan	2%
monitor requirements/specification compliance	12%
aid in decision making	2%
identify problems	4%
ability to be proactive	2%
track SPRs	2%
phase/milestone completion	4%
provide information to outside agencies	2%
monitor software performance	6%
additional source of data	2%
cost purposes	10%



manpower	2%
customer's needs	2%
quality	2%
size	2%

#### Question 20--Transform

##### engineers

documentation review	19%
compare requirements to contractor performance	6%
evaluate deviations	1%
evaluate manpower	1%
evaluate completeness of work	1%
software testing	12%
interface with other teams	12%
evaluate legal aspects	2%
requirements	10%
contract management/monitoring	2%
develop/implement policy	6%
technical assistance	8%
identify problems	2%
maintainability	2%
software reviews	2%
work with contractor	4%
strategic plans	2%
software design	4%
software develop	2%
computer security	2%

##### managers

responsible for cost	20%
responsible for schedule	20%
responsible for performance	5%
work with other teams	10%
responsible for product	5%
responsible for delivery/fielding	10%
responsible for quality	5%
responsible for requirements	5%
ECP evaluation	5%
monitor indicators	5%
reliability improvements	5%
chair ICWG	5%

#### Question 11--Transform

pass information to PM/up chain/ to other teams	21%
convert to verbal update	2%
review data	4%
crosscheck contractor report	2%
trend analysis/prediction	10%

track progress	8%
generate actions	19%
status to outside agency	4%
identify missing requirements	6%
identify problems	10%
helps contractor	4%
verify hardware compatibility	2%
award fee criteria	4%
prioritize problems	2%
spotlight risks	2%
elevate problems	2%

#### Question 23--Voice of the Customer

requirements traceability	7%
testing information	7%
workbook on indicators	7%
formal SPRs	7%
formal complexity metric	7%
historical database	7%
software performance data	7%
results from quality reviews	7%
information on contracting for indicators	13%
correct indicators	7%
indicator training	7%
new programs need information on	
complexity	7%
documentation information	7%
integrated schedule	7%

#### Question 25--Voice of the Customer

more important for contractor	4%
gear to needs/phases of program	12%
book/workshop on metrics	8%
do not rely on indicators exclusively	8%
knowledgeable people to interpret metrics	8%
standard set does not make sense	4%
more information on how to use indicators	12%
automated tools/listings of metrics	4%
have indicators for a reason-not just to	
have them	4%
balance cost with need for indicators	4%
manpower indicator	4%
maintainability indicator	4%
requirements indicator	4%
general lack of knowledge on indicators	4%
accuracy of data is important	12%
indicators should be on every contract	4%
trouble defining process compliance	
indicator	4%

#### Question 24--Outputs

meeting minutes	3%
verbal updates	19%
identify problem areas	3%
activity/status report	6%
briefings	24%
program reviews	8%
CPARS input	2%
memos	8%
trip reports	16%
watch list	2%
indicator data	2%
software schedule	2%
newsletter	2%
white papers	2%
E-mails	3%

#### Question 12B--Research Question

<u>no</u>	
more for contractor than for SPO	17%
good software development process	33%
not being used at program level	33%
get data from other sources	17%

<u>yes</u>	
gets attention	16%
identify problems early, minimize schedule impact	16%
helps compare actual vs predicted	11%
made code better	6%
show inconsistencies	11%
improve visibility into process	21%
part of development record	6%
helps engineer, therefore helps program	6%
quantifies activities	6%
force contractor to plan	6%

#### Question 9--Research Question

800-43 as adapted by contractor	37%
contractor internal indicators	26%
developed during dem/val	5%
SPO proposed set	5%
SPO policy	11%
literature review baseline tailored by contractor	5%
experience	5%

# Question 8--Research Question

## no

cost	18%
provide no influence on contractor	9%
no emphasis/knowledge of indicators at contract award time	55%
Navy DID used	9%
good contractor relationship	9%

## yes

contractor saw need	22%
convinced contractor indicators would help track plan and schedule	11%
basis for award fee	11%
contractor wanted to satisfy SPO	11%
"no problem"	44%

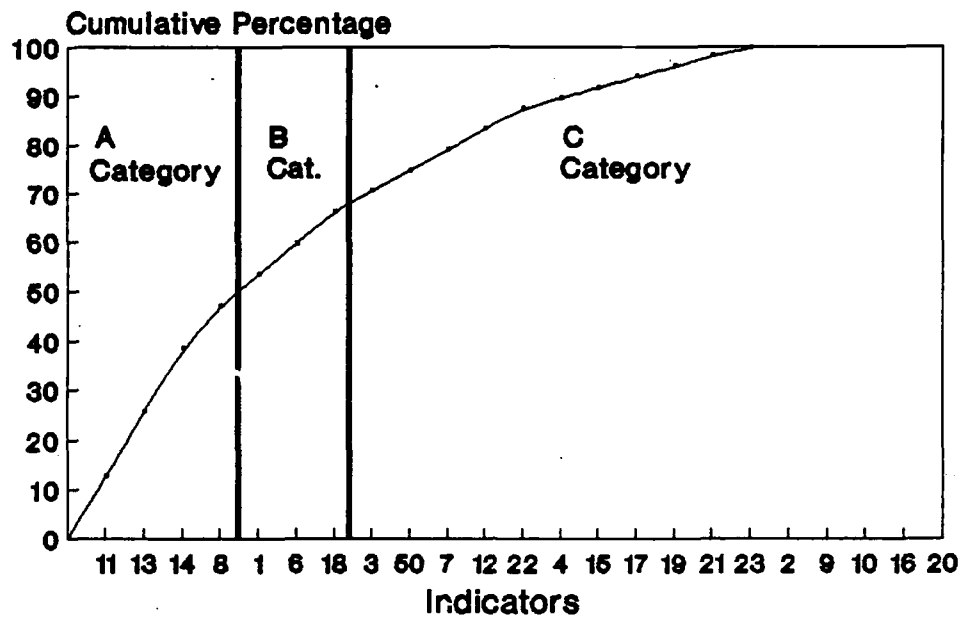
# Appendix F Results of Indicator and Area Sheets

## SOFTWARE DEVELOPMENT AREAS

ctrl #	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12
1	4	5	3	5	4	5	4	4	4	5	4	4
2A	3	4	3	5	3	4	2	5	5	4	4	3
2B	4	4	3	5	4	4	4	5	5	4	4	5
3	5	4	1	4	3	4	2	3	2	3	4	3
4	4	3	2	5	4	4	3	4	5	4	4	4
5	4	4	4	5	3	4	3	5	5	4	4	3
6	4	4	4	5	4	5	3	5	5	4	4	5
7	3	3	4	5	3	4	2	5	5	3	4	1
8	4	3	2	4	4	2	2	2	4	4	5	3
9A	5	5	5	4	2	4	4	4	5	4	2	3
9B	4	4	3	5	4	5	4	5	5	4	4	4
10	4	4	4	4	3	4	4	4	4	4	4	5
11	5	5	4	4	4	4	4	5	5	4	5	5
12A	5	4	5	5	5	5	5	4	5	5	5	5
12B	5	5	5	5	5	5	5	5	5	5	5	5
13	2	4	2	5	4	5	4	5	4	5	4	3
14	4	5	4	5	5	5	4	5	5	5	5	5
15	5	3	5	4	4	5	5	4	5	5	4	5
16	4	5	4	5	4	5	4	4	4	5	4	4
17	4	5	4	5	4	5	3	4	5	5	4	5
18	5	2	2	3	2	4	4	4	5	3	5	3
19	5	3	3	5	5	2	2	5	3	2	5	3
20	4	4	4	4	3	4	3	4	4	3	2	4
21	4	4	3	4	4	4	3	4	4	4	4	4
22	5	2	4	3	4	3	5	4	4	2	4	2
23	5	4	5	5	5	5	4	4	5	4	3	5
24	3	5	3	5	5		4	5	5	5	5	5
25	5	4	3	5	5	4	3	5	5	4	5	3
26	5	5	3	3	3	4	4	4	4	4	3	3
27	5	2	4	5	3	5	3	5	4	4	3	3
28	5	3	4	4	4	4	4	3	5	4	4	5
29	4	3	3	4	3	5	3	5	5	5	5	4
30	1	3	3	5	4	3	4	5	5	3	4	5
31	5	3	5	5	3	4	4	4	5	4	4	4

chl #	1	2A	2B	3	4	5	6	7	8	9A	9B	10	11	12A	12B	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
i1	3	1	4	5	5	4	0	4	4	3	0	4	0	0	0	5	5	0	4	4	0	0	0	4	3	0	0	4	5	4	4	0	0	4	
i2	4	0	3	0	0	3	0	0	0	0	0	0	0	0	0	4	4	0	0	0	0	0	0	3	0	3	0	0	4	4	4	0	0	0	
i3	5	3	4	5	4	5	4	4	5	4	0	4	0	0	5	5	4	5	5	5	4	4	4	4	5	5	5	4	5	5	4	4	4	4	
i12	4	1	3	3	3	4	3	3	0	5	0	0	0	0	5	0	4	5	5	0	4	3	3	0	2	4	0	4	5	5	4	2	5	4	4
i11	3	5	3	3	4	5	4	0	5	4	5	4	0	0	5	4	4	4	4	4	3	3	0	4	4	0	4	5	5	5	4	5	2	5	5
i10	4	0	3	3	0	0	3	4	0	5	5	0	3	0	4	4	2	0	0	4	0	0	0	3	4	0	0	0	0	0	0	3	0	0	0
i9	5	1	4	4	4	4	3	4	4	4	5	4	0	0	4	3	2	4	4	4	5	5	4	4	5	3	5	4	4	4	3	5	3	4	
i8	5	1	5	4	4	0	5	4	4	5	0	4	0	0	5	4	5	5	4	5	0	0	5	2	3	4	5	5	5	5	5	4	5	3	5
i7	5	3	4	0	0	3	3	4	0	5	0	0	0	0	5	5	4	5	5	4	0	4	5	0	4	5	0	4	5	0	0	0	0	4	4
i6	4	5	4	4	5	3	3	3	3	4	3	4	0	0	5	2	3	0	0	4	4	5	2	0	2	5	3	5	5	4	4	0	3	4	4
i5	5	0	0	4	0	0	3	3	4	0	0	0	0	0	4	3	5	0	5	5	4	0	4	2	0	4	4	5	5	4	4	4	0	0	0
i4	5	0	5	0	0	3	5	5	0	5	5	0	0	0	4	4	5	5	4	4	0	4	2	3	5	4	5	5	4	4	4	4	4	0	0
i3	5	3	3	3	4	4	3	5	4	4	0	0	0	0	5	4	4	5	4	5	0	4	2	2	4	5	5	4	4	4	4	4	4	3	4
i2	4	0	3	0	0	3	3	0	0	0	0	0	0	0	4	3	4	4	5	5	0	0	0	3	4	3	0	0	4	4	4	0	0	0	0
i1	3	1	4	5	5	4	0	4	4	3	0	4	0	0	5	5	5	5	5	4	4	3	0	4	4	3	0	4	5	5	4	4	2	5	4
i11	3	5	4	0	5	4	0	4	4	5	4	0	0	0	5	4	4	4	4	4	0	0	0	4	4	0	0	0	4	4	4	0	0	0	0
i10	4	0	3	3	0	0	3	4	0	5	5	0	3	0	4	4	2	0	0	4	0	0	0	3	4	0	0	0	0	0	0	3	0	0	0
i9	5	1	4	4	4	4	3	4	4	4	5	4	0	0	4</																				

## Pareto Analysis for Indicators (Managers' Responses)



### A Category

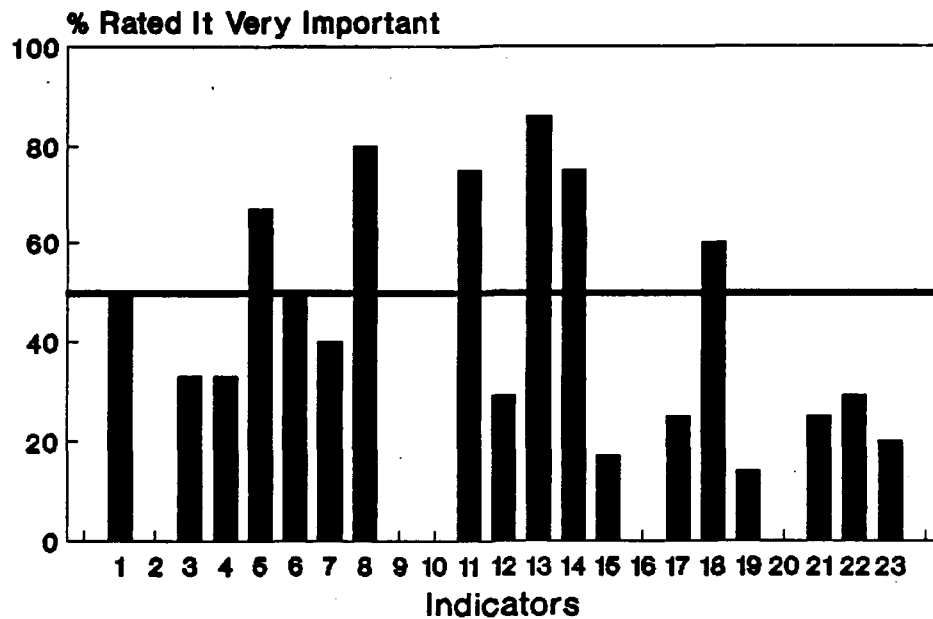
11 - Memory Utilization

13 - Throughput

14 - Requirements Traceability

8 - Requirements Definition  
and Design Stability

## Chosen By Majority (Managers' Responses)

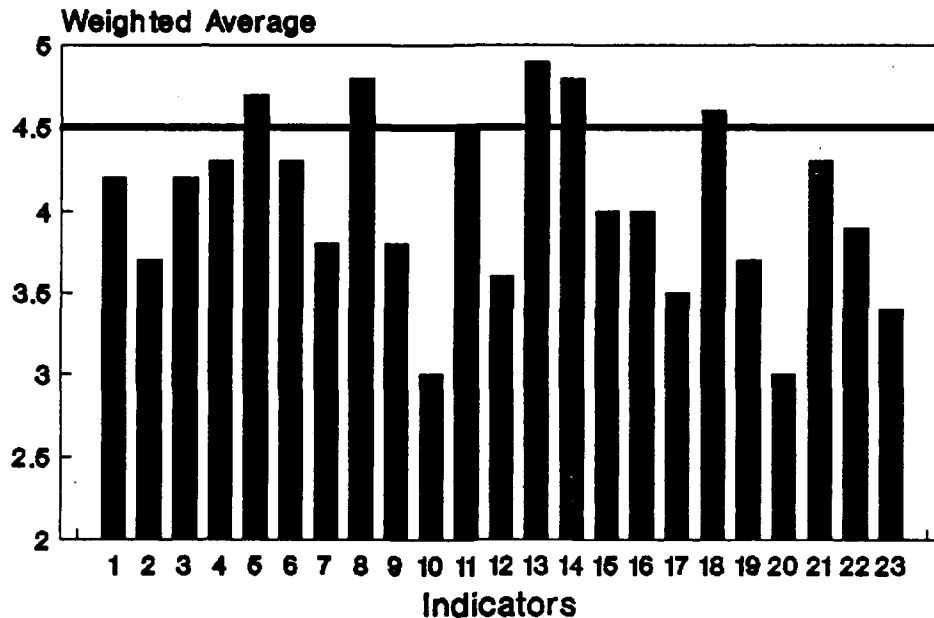


### Indicators Receiving Majority Responses

- 13 - Throughput
- 8 - Requirements Definition  
and Design Stability
- 11 - Memory Utilization
- 14 - Requirements Traceability
- 5 - Test Sufficiency
- 18 - Requirements Compliance



### Weighted Average (Managers' Responses)



#### Indicators With Weighted Averages > 4.5

13 - Throughput

8 - Requirements Definition  
and Design Stability

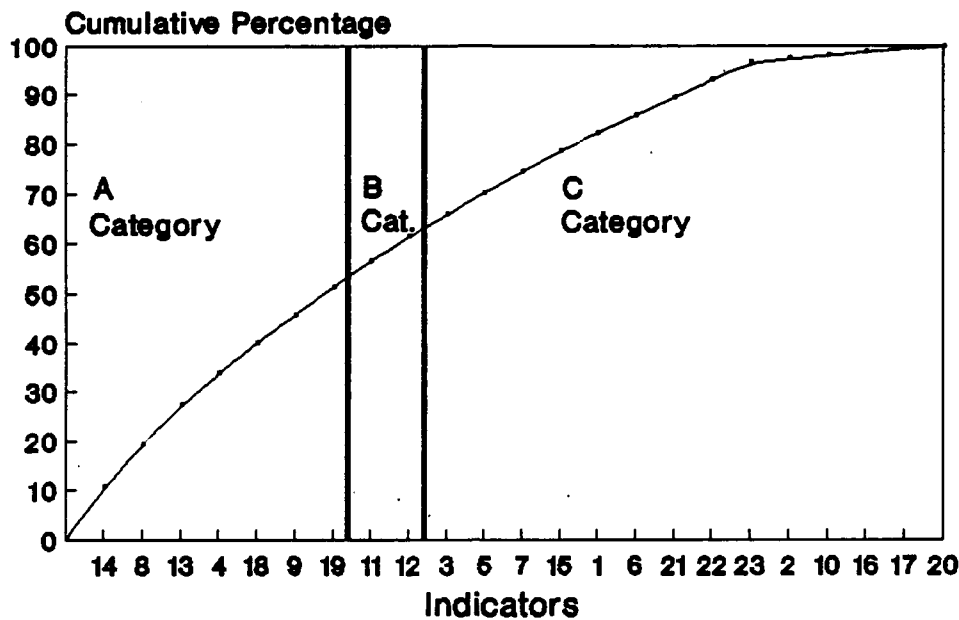
14 - Requirements Traceability

5 - Test Sufficiency

18 - Requirements Compliance

11 - Memory Utilization

## Pareto Analysis for Indicators (Engineers' Responses)



### A Category

14 - Requirements Traceability

8 - Requirements Definition  
and Design Stability

13 - Throughput

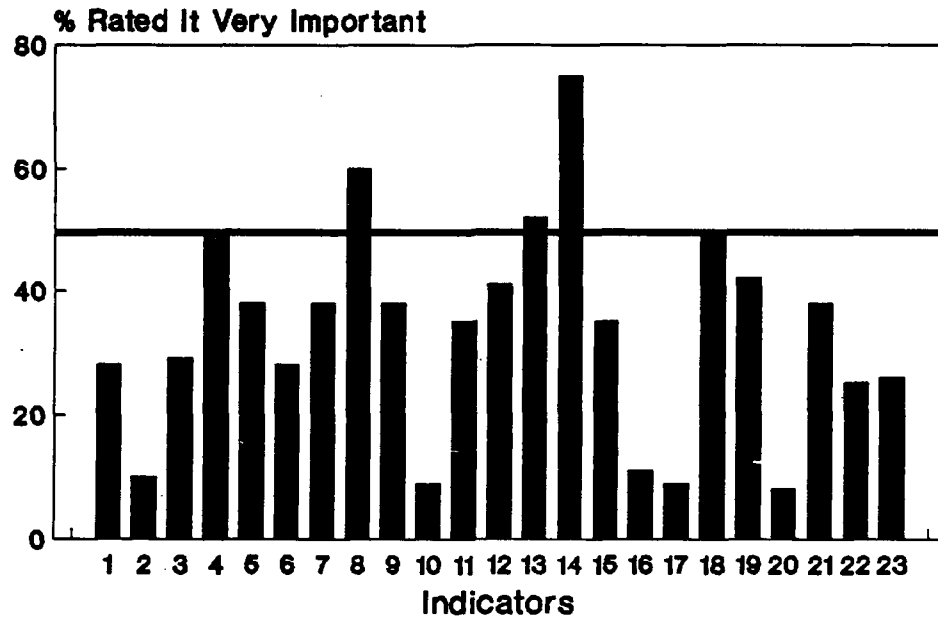
4 - Test Coverage

18 - Requirements Compliance

9 - Schedule Progress

19 - Schedule Progress -  
Development and Test

## Chosen By Majority (Engineers' Responses)



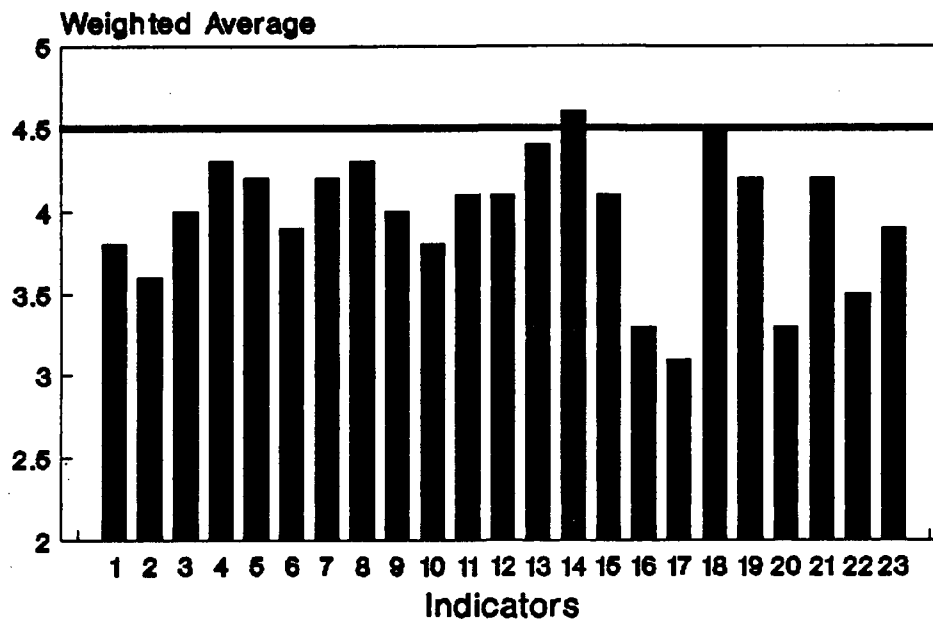
### Indicators Receiving Majority Responses

14 - Requirements Traceability

8 - Requirements Definition  
and Design Stability

13 - Throughput

### Weighted Average (Engineers' Responses)

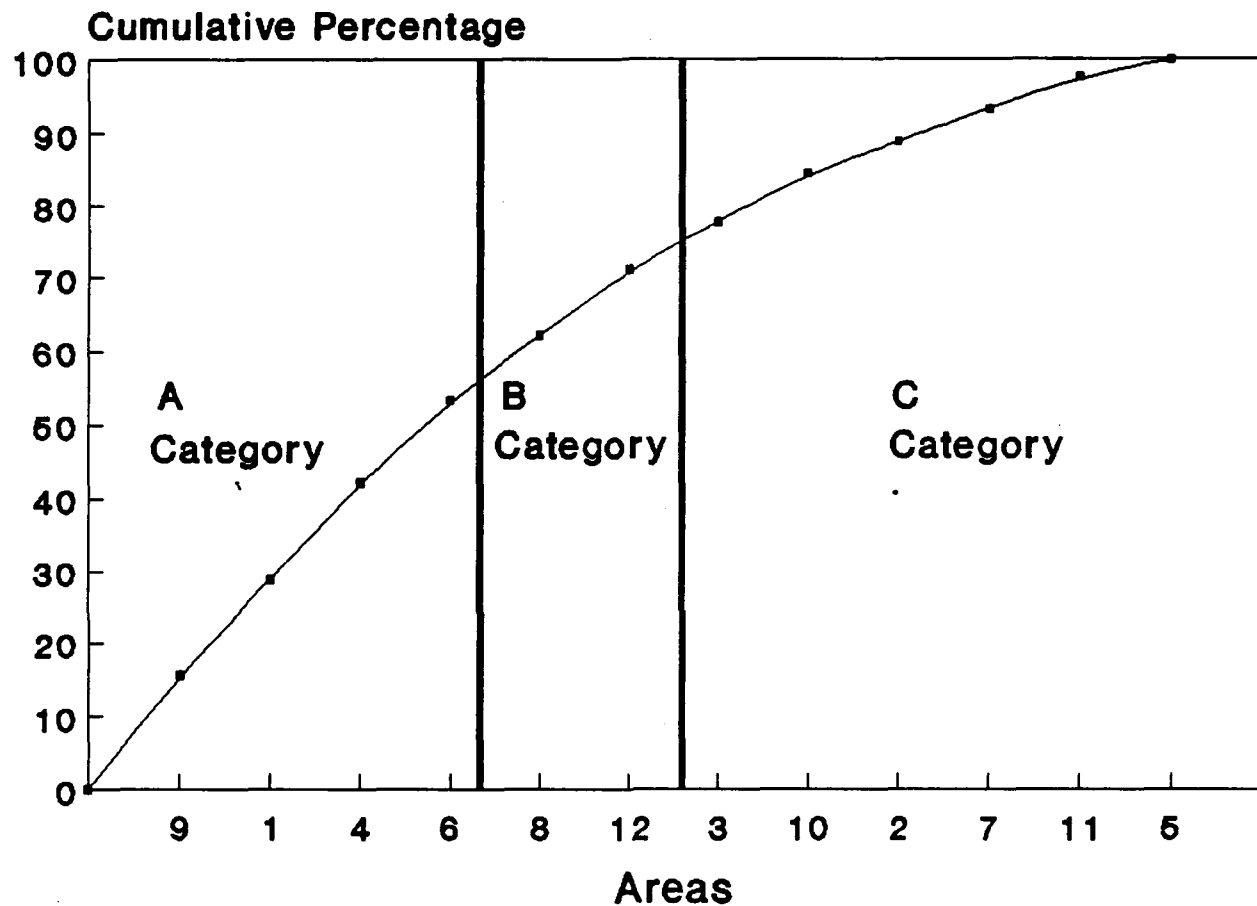


#### Indicators With Weighted Averages > 4.5

14 - Requirements Traceability

18 - Requirements Compliance

## Pareto Analysis for Areas (Managers' Responses)



### A Category

- 9 - Requirements
- 1 - Schedule
- 4 - Software Performance
- 6 - Quality

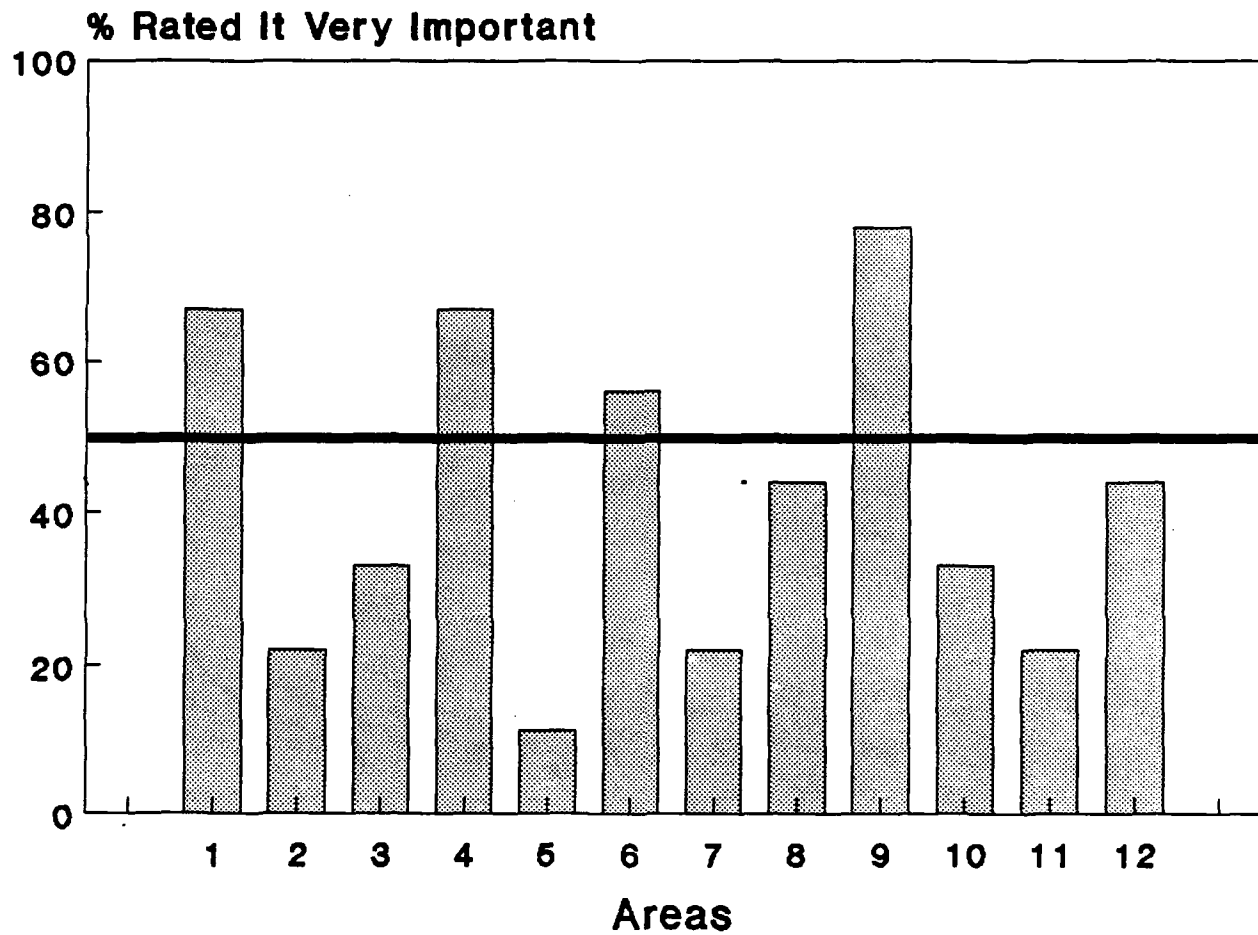
### B Category

- 8 - Testing
- 12 - Software Process Consistency

### C Category

- 3 - Cost
- 10 - Maintainability
- 2 - Reliability
- 7 - Manpower
- 11 - Documentation
- 5 - Software Characteristics

## Chosen By Majority (Managers' Responses)



### Areas Receiving Majority Responses

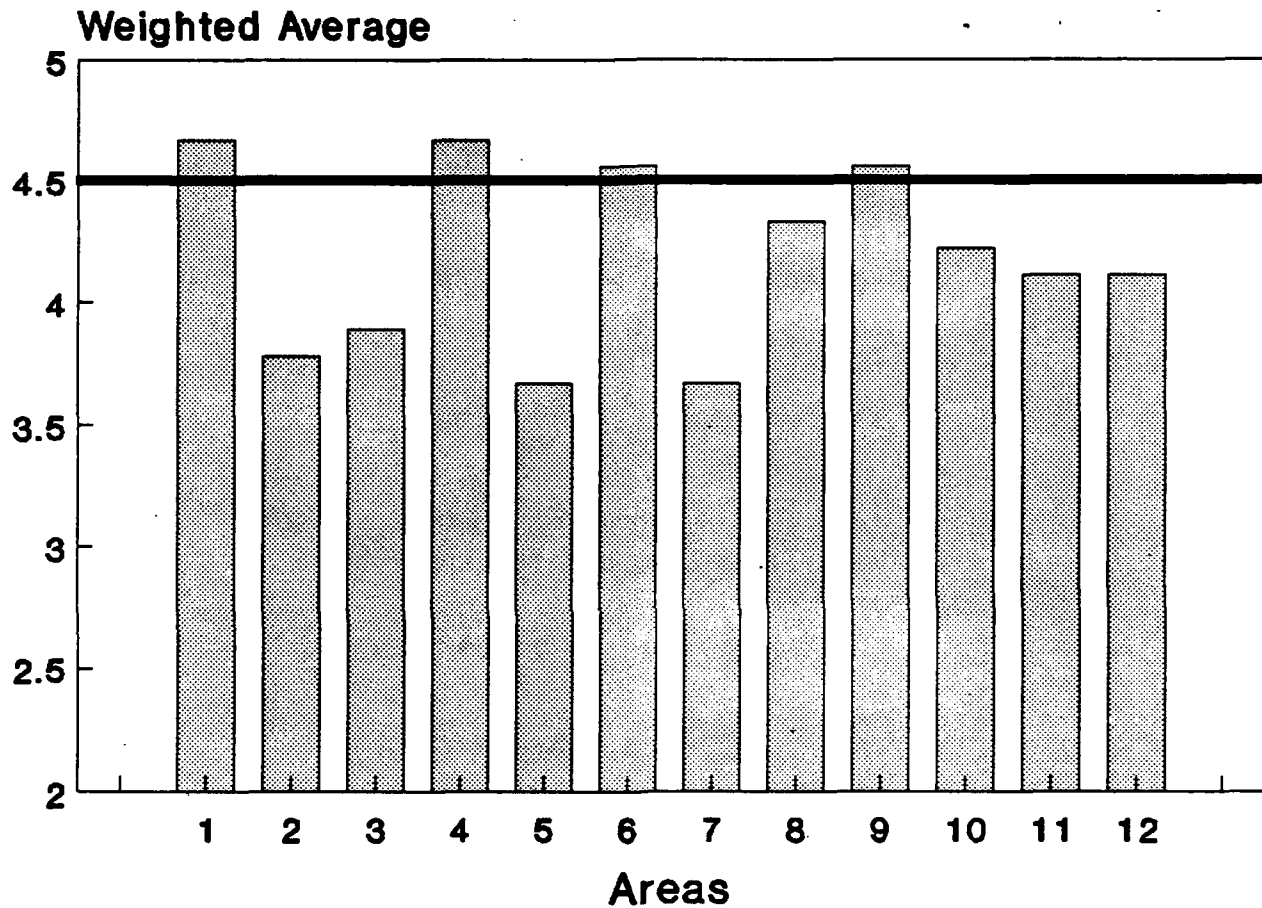
9 - Requirements

1 - Schedule

4 - Software Performance

6 - Quality

## Weighted Average (Managers' Responses)



### Areas With Weighted Averages > 4.5

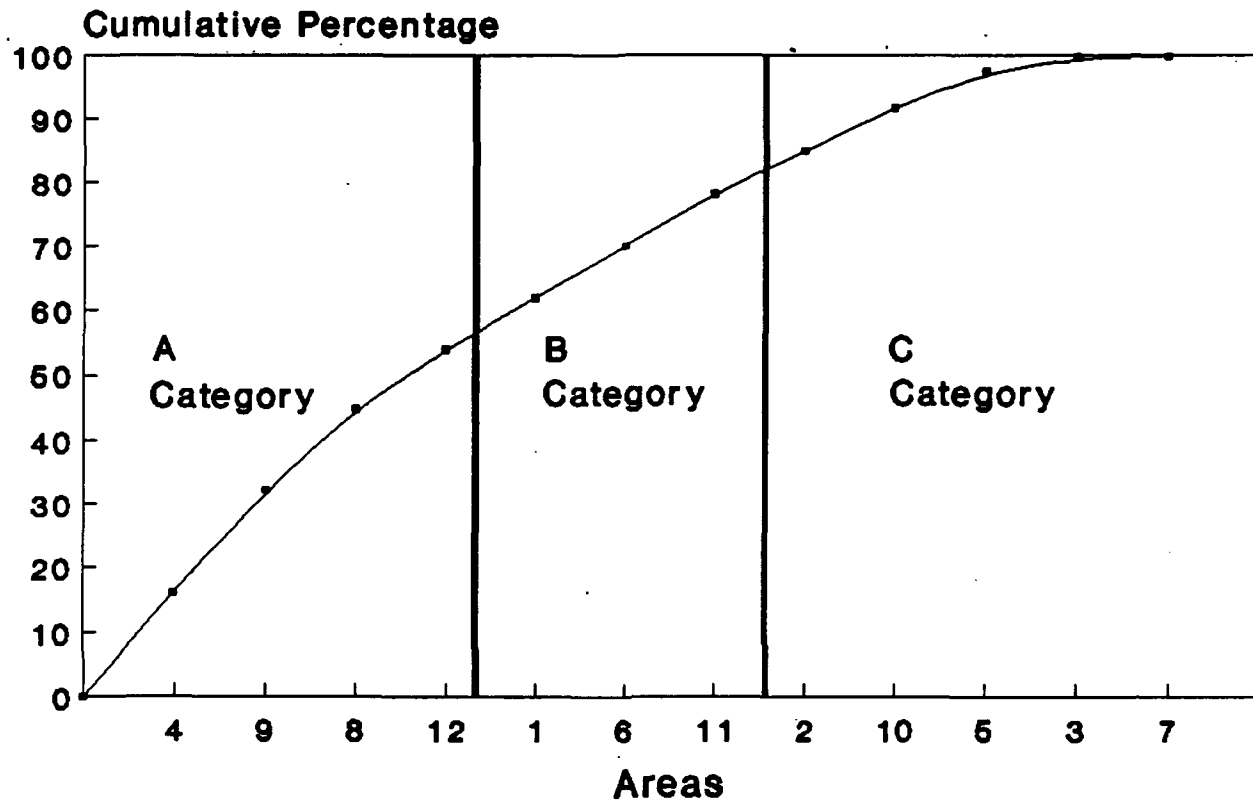
1 - Schedule

4 - Software Performance

6 - Quality

9 - Requirements

## Pareto Analysis for Areas (Engineers' Responses)



### A Category

- 4 - Software Performance
- 9 - Requirements
- 8 - Testing
- 12 - Software Process Consistency

### B Category

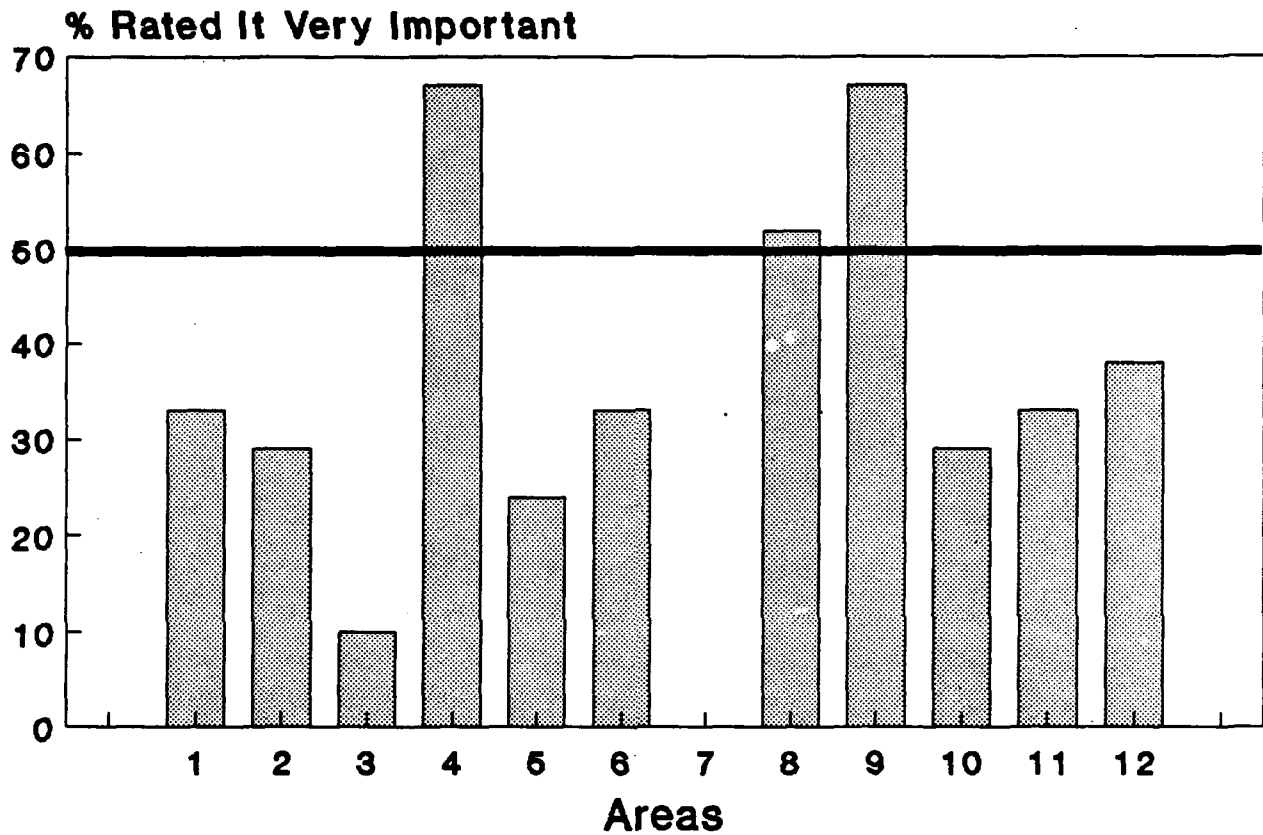
- 1 - Schedule
- 6 - Quality
- 11 - Documentation

### C Category

- 2 - Reliability
- 10 - Maintainability
- 5 - Software Characteristics
- 3 - Cost
- 7 - Manpower



## Chosen By Majority (Engineers' Responses)



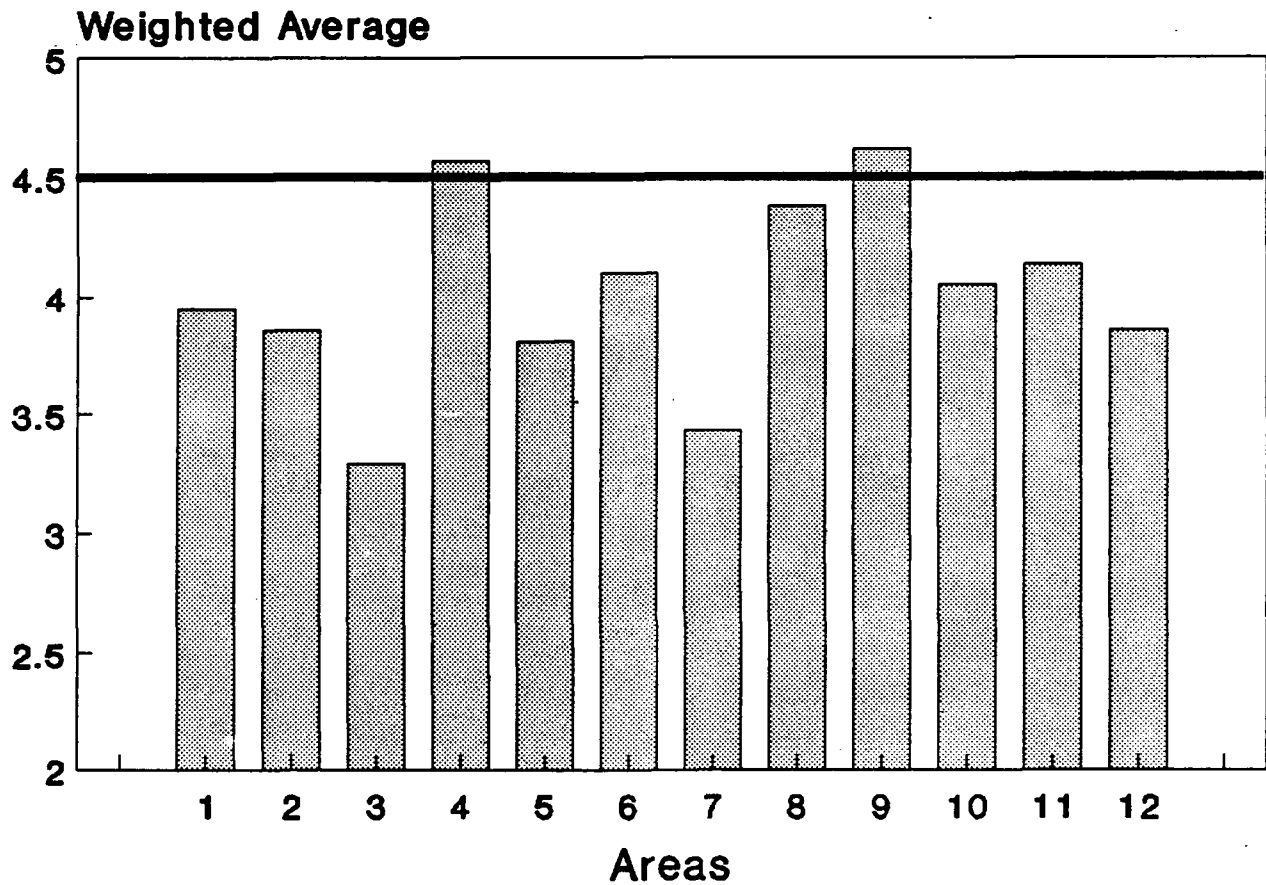
### Areas Receiving Majority Responses

4 - Software Performance

9 - Requirements

8 - Testing

## Weighted Average (Engineers' Responses)



### Areas With Weighted Averages > 4.5

9 - Requirements

4 - Software Performance

**Appendix G.**

**DRAFT**

**Software Indicator  
Handbook**

**For**

**Computer Resources Branch,  
Directorate for Avionics Engineering,  
Aeronautical Systems Center  
(ASC/ENASC)**

**Draft**

## Table of Contents

Section 1.0	Why Do You Need Indicators
Section 2.0	Indicator Process at ASC
Section 3.0	Problems and Solutions
Section 4.0	The Standard Set of Indicators
Section 5.0	Other Sources of Indicator Information

## Section 1.0 Why Do You Need Indicators

The first step to improve management of a process is to provide data to allow visibility into what is happening within that process. As Lord Kelvin stated:

"When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind..."

Likewise, William Sherkenbach states the following concerning Dr. W. Edwards Deming:

"Dr. Deming has said many times that management's job is prediction. This is because prediction increases knowledge, and knowledge is a prerequisite for action. When the predictions and the actions coincide, this increases wisdom. In order to predict, you must have data."

Software indicators, or metrics, provide such data for the software development process. This is emphasized by DOD directing in DODD 5000.2, Defense Acquisition Program Procedures, that each service implement software indicators for its acquisition efforts. In addition, MIL-STD-1803, Software Development Integrity Program, requires the contractor to adopt software management indicators.

Air Force Systems Command (AFSC) has published pamphlets AFSCP 800-43, Software Management Indicators, and AFSCP 800-14, Software Quality Indicators, that describe software indicators that may be used by organizations involved with the acquisition of software. However, these pamphlets are only guides and the actual selection and implementation of software indicators is up to each software manager and engineer.

## Section 2.0 Indicator Process at ASC

### 2.1 Where to Start

The first place to start is with a visit to ASC/ENASC. ASC/ENASC maintains a database of currently available indicators. In addition, they maintain a database of the indicator data gathered from programs throughout all of ASC. They use this data to generate better program estimates and improve cost models.

With their knowledge of indicators and their uses, they can help in tailoring and selecting an indicator set for your program. They can also share lessons learned from other programs within ASC to help your effort be successful.

### 2.2 How to Select a Set of Indicators

2.2.1 The Standard Set of Indicators. Through research efforts of the Air Force Institute of Technology, supported by ASC/ENASC, a standard set of software indicators has been developed based on the needs identified by software managers and engineers. This standard set of indicators has provided a good basis from which to develop a set of indicators for your program.

2.2.2 Results from the Software Development Capability/Capacity Review (SDCCR). When evaluating a contractor's capability, areas can be identified that may need increased management attention. These areas may prompt you to add indicators to the standard set to provide visibility into these areas.

2.2.3 MIL-STD-1803, Software Development Integrity Program. Mil-Std-1803 provides a list of requirements for the contractor to adhere to. In order to meet these requirements, activities are established in order to assure that software is developed correctly and milestones are met. These activities are documented in an Integrated Master Plan and Integrated Master Schedule. The activities can provide points to measure against which can be used to establish software indicators.

### 2.3 How to Contract for Indicators

2.3.1 Sources for Statement of Work (SOW) and Contract Data Requirements List Examples (CDRL). AFSCP 800-43, Software Management Indicators, and ESD-TR-88-01, Software Management Metrics provide guidance and examples for getting indicators on contract.

2.3.2 Items to Consider. To ensure that the indicator program is successful, you have to ensure that the contractor has a good software indicator program. To do this, evaluation

criteria should be incorporated into the source selection and contractor proposal evaluations. Again, the SDCCR can provide a considerable amount of insight into the contractor's software development program.

Other things to consider are the frequency that you receive indicator data and the method by which the contractor delivers it to you. The best solution is to specify that the contractor allow you direct access to the software indicator database. This gives you the opportunity to view the data anytime you want to. An alternative is to have the contractor fax the information to you which cuts down on delivery time. The worst situation is to incorporate the data requirements into a monthly status report where it may lose focus on its importance resulting in data that is delivered too late for you to act on.

One other item to keep in mind is delivery of data to ASC/ENASC. To maintain the database and continue to improve on estimates and cost models, ASC/ENASC needs to collect indicator information from all programs within ASC. You can either provide a copy of your data to ASC/ENASC or have the contract send them one by putting the requirement on the DD Form 1423 that has the distribution list for the data item. When you initially contact ASC/ENASC, this is an item that you can discuss with them to work out an arrangement that is best for your program.

2.3.3 Validating the Indicator Data. One thing to keep in mind is the issue of how good is the data you are getting. When accomplishing the SDCCR, it is good to check out the contractor's measuring system and determine how it is validated. During the program, you should also continue to validate the data the contractor sends you. The local Defense Plant Representative Office can aid you in validating that the data the contractor is submitting is valid. Crosschecks can also be accomplished with cost/schedule status reports. When on plant visits or attending design and program reviews can serve as an additional opportunity for ensuring the accuracy of the indicator data.

## Section 3.0 Problems and Solutions

### 3.1 Identified Problems

A survey accomplished by the Air Force Institute of Technology revealed number problems associated with the use of software indicators. These problems were identified as: 1) not getting indicator data in a timely fashion, 2) indicators not meeting the needs of the respondent by either being for a different function or for use by higher-level management, 3) indicators providing data that could have been obtained through other sources, 4) believing that indicators are more beneficial to the contractor than the program office, 5) the software developer refusing to provide the data, 6) not tailoring the indicators for specific phases of the program, 7) identifying the need for knowledgeable people to interpret metrics, 8) a lack of knowledge about indicators themselves, 9) a lack of management involvement, 10) the cost of putting indicators on contract, 11) the lack of emphasis placed on indicators at the time of contract award, and 12) ambiguous definitions of indicators. Recommendations concerning these problems will be made in the next section.

### 3.2 Solution to Problems

Each of problems will be addressed separately with recommendations are ongoing actions that will resolve the problems. The recommendations are based on the assumption that the software indicators are on contract.

The solution to not getting indicators in a timely fashion may be resolved by two different methods: change reporting period or change method of delivery. The first method can be adjusted to meet the needs of the manager or engineer but has the drawback of potentially increasing costs as shorter delivery times are mandated. The second can be accomplished several ways. One way is to have direct access to the software developer's database so that information can be obtained at any time. Another approach would be to have the software developer transmit a facsimile of the indicators which would certainly speed up the mailing process. The main point to be made is to avoid tying the delivery of indicators to a standard data item, like a monthly status report, where indicators are not the primary focus of the data item.

The solution to indicators not meeting the needs of the respondents should be resolved with the standard set of indicators in Section 4.0 of this handbook as as these were derived from the stated needs of the individuals interviewed. Though a standard set will not satisfy all needs for every program, it certainly should provide a firm basis for supporting management of software development efforts within ASC.



The solution to the problem of indicators providing redundant information that could have been obtained from other sources is to ensure that no duplication exists between various sources of information available to managers and engineers. The standard set of indicators should certainly provide information necessary to the management of the software development effort. They also provide a all the necessary information in a single product which allows for better assessment of the impacts of the interactions of the various core areas identified. If there are other sources of information available, the benefit of those sources over the advantages of having a standard set of indicators must be weighed carefully to determine which provides the most benefit for the cost.

The solution to the premise that indicators are only beneficial to the software developer and not managers and engineers can be resolved by noting that not all problems identified by indicators are due to the actions of the software developer and some may be directly linked to the actions of the program office. This is not to say that the software developer should not be concerned about indicators, but there should be a joint effort between the software developer and the program office to identify problems and make improvements to resolve them. This will result in a better process and therefore a better product.

The solution to the software developer refusing to provide the data can be resolved by placing the task of providing indicator data on contract. Though this may not totally resolve the reluctance of the software developer to provide data, it provides a vehicle for the program office to obtain the information by making it a binding agreement between the two parties.

The solution to not tailoring indicators to the different phases of the software development effort is in the guidance provided by the procedures to implement the standard set of software indicators. The procedures show a matrix of the standard indicators and program reviews and the phasing of the indicators in relation to those reviews.

A solution to the problem of not having knowledgeable people to interpret indicator data cannot be provided. Attempts have been made to provide details of how to use the standard indicators and interpret their data.

The solution to the problem of a lack of knowledge of software indicators can be alleviated by contacting ASC/ENASC to obtain further information about indicators. This handbook should also be educational to managers and engineers as it provides information on indicators and their uses.

The solution to the lack of management involvement needs to be addressed by top management. Top management must become involved and support data gathering efforts for them to be successful. If top management becomes involved with and supports the effort, managers throughout the organization will do the same. With the recent publication of ASDP 700-8, ASD Metrics Handbook, and the award to ASC of the Quality Improvement Prototype Award, there appears to be an increased management focus on indicators which may mitigate this problem.

A solution to the cost of putting indicators on contract cannot be provided. The cost of putting indicators on contract has been minimized by the development of a core set of indicators based on those areas identified as very important to software managers, engineers, and ASC/ENASC.

There may already be a solution in place to the problem of a lack of emphasis on indicators resulting in indicators not being placed on contract. Mitigating factors in the lack of emphasis would be the ASDP 700-8, "ASD Metrics Handbook 30 Apr 92" and this research effort. There is also a marked swing in later contracts to the effort to put indicators on contract, as evidenced by the average year of contract award.

The solution to the problem of ambiguous definitions of indicators may be resolved by the development of the indicator handbook. Attempts were made to define the indicators to the greatest extent possible. Recommendations can be made to ASC/ENASC to work with the computer resource focal points to further add definition to the standard set of indicators and clear up any ambiguities.

## Section 4.0 The Standard Set of Indicators

The standard set of software indicators is divided into four areas: schedule, cost, requirements, and software performance. The indicators within each area will be discussed.

The indicators span all phases of the software development cycle as evident by the following matrix.

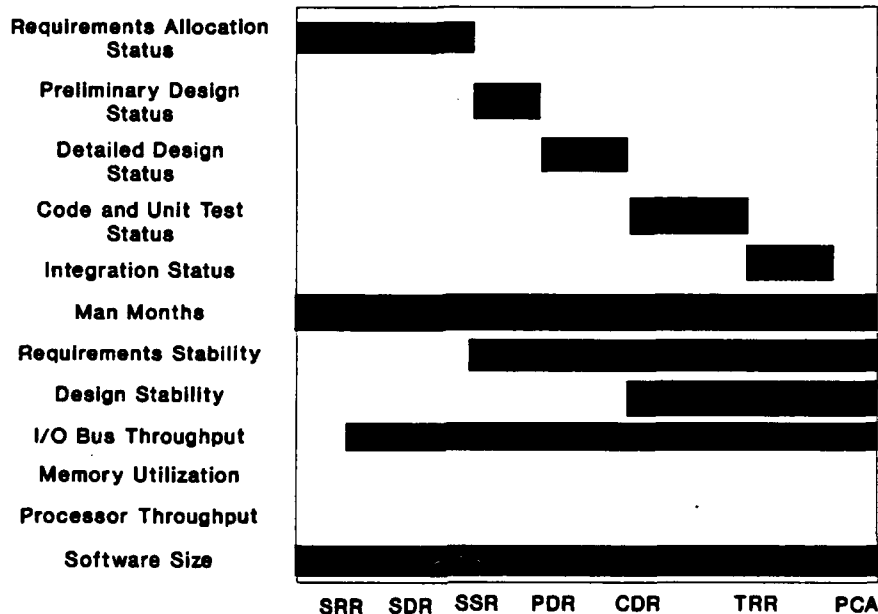


Figure 4-2. Phasing of Software Indicators

4.1 Schedule Indicators. There are six indicators within the schedule area are: Requirements Allocation Status, Preliminary Design Status, Detailed Design Status, Code and Unit Test Status, and Integration Status.

### 4.1.1 Resource Allocation Status

4.1.1.1 Purpose. The purpose of the Resource Allocation Status indicator is to track the progress of allocating requirements to the software CSCIs through the development of the Software Requirements Specification (SRS) and Interface Requirements Specification (IRS).

4.1.1.2 Frequency of Update. This indicator will be updated monthly from the System Requirements Review until approval of the Software Requirements Specifications (SRS) and Interface Requirements Specifications (IRS).

4.1.1.3 Update Criteria. Data for updates will be obtained by weekly inspections of the completed portions of the SRS and IRS.

4.1.1.4 Indicator Inputs. The indicator is constructed from the following information:

a. Number of Requirements Allocated to the CSCIs: Data is obtained from the requirements in the System/Segment Design Document to be allocated to the CSCI.

b. Planned Number of Requirements Allocated: Data is obtained from estimated number of requirements each reporting period expected to complete requirements analysis and be incorporated in the SRS and IRS.

c. Actual Number of Requirements Allocated: Data is obtained from actual number of requirements documented in the SRS IRS for each reporting period.

d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations.

4.1.1.5 An Example of the Indicator. An example of the Requirements Allocation Status indicator is shown in Figure 4-1.

4.1.1.6 Using the Indicator. The indicator provides information concerning how well the resource allocation activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the resource allocation effort while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the resource allocation task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.1.1.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

# CSCI (Name of CSCI) Requirements Allocation Status

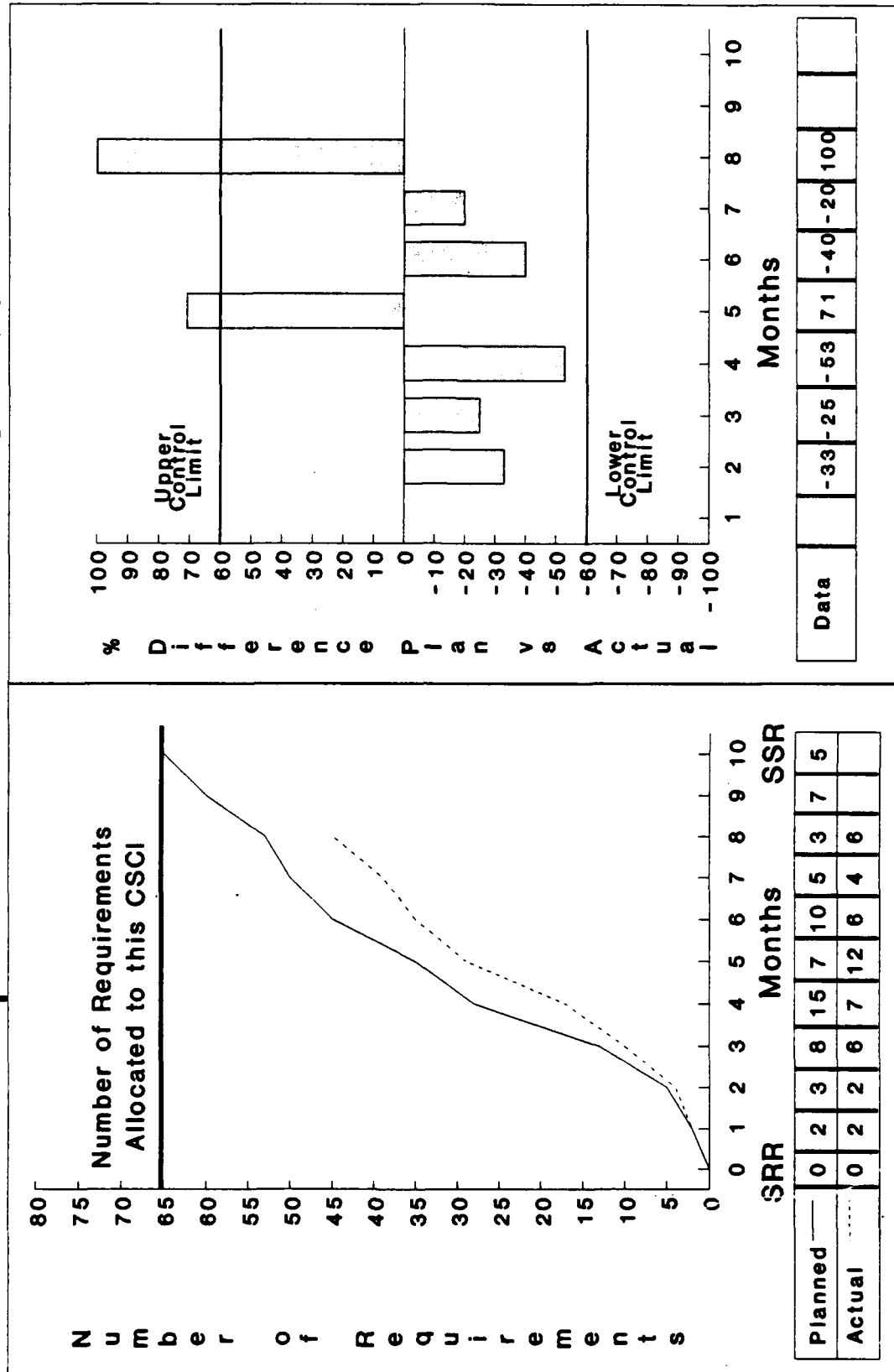


Figure 4-1 Example of Requirements Allocation Status Indicator

- a. Have separate indicators for the SRS and IRS
- b. Breakout Planned and Actual data for specific sections of the SRS and IRS (Sections 3 and 4).
- c. Change frequency of updates to meet your needs.
- d. Change time period of indicator to meet your needs.

#### 4.1.2 Preliminary Design Status

4.1.2.1 Purpose. The purpose of the Preliminary Design Status indicator is to track the progress of translating the requirements into design for the software CSCs through the development of the Software Design Document (SDD) and Interface Design Document (IDD).

4.1.2.2 Frequency of Update. This indicator will be updated monthly from the Software Specification Review until the Preliminary Design Review.

4.1.2.3 Update Criteria. Data for updates will be obtained by weekly inspections of the completed portions of the Software Design Document (SDD) and the Interface Design Document (IDD).

4.1.2.4 Indicator Inputs. The indicator is constructed from the following information:

a. Number of Requirements Allocated to the CSCIs: Data is obtained from the number of requirements in the SRS for the CSCI.

b. Planned Number of Designs Documented: Data is obtained from the estimated number of requirements each reporting period expected to complete preliminary design and be incorporated into the SDD and IDD.

c. Actual Number of Designs Documented: Data is obtained from actual number of requirements documented in the SDD and IDD for each reporting period.

d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations.

4.1.2.5 An Example of the Indicator. An example of the Preliminary Design Status indicator is shown in Figure 4-2.

4.1.2.6 Using the Indicator. The indicator provides information concerning how well the preliminary design activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the preliminary design effort while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the preliminary design task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.1.2.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Have separate indicators for the SDD and IDD
- b. Breakout Planned and Actual data for specific sections of the SDD and IDD (Sections 3 and 4).
- c. Change frequency of updates to meet your needs.
- d. Change time period of indicator to meet your needs.

### 4.1.3 Detailed Design Status

4.1.3.1 Purpose. The purpose of the Detailed Design Status indicator is to track the progress of translating the requirements into design for the software CSUs.

4.1.3.2 Frequency of Update. This indicator will be updated monthly from the Preliminary Design Review until the Critical Design Review.

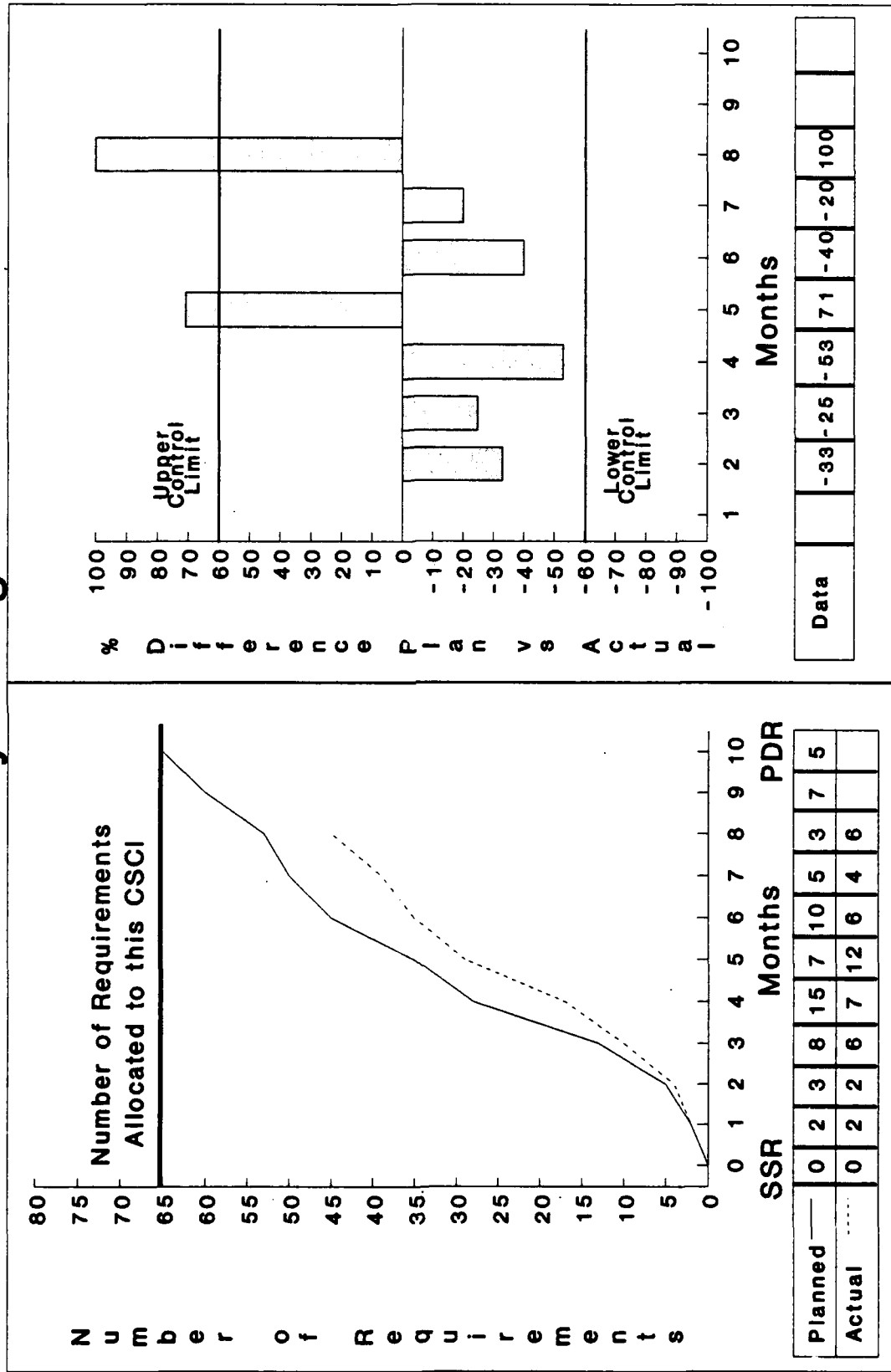
4.1.3.3 Update Criteria. Data for updates will be obtained by weekly inspections to determine the completeness of the detailed design for the CSUs.

4.1.3.4 Indicator Inputs. The indicator is constructed from the following information:

- a. Total Number of CSUs in the CSCI: Data is obtained from estimates or actual numbers if known.

Figure 4-2 Example of Preliminary Design Status Indicator

# CSCI (Name of CSCI) Preliminary Design Status



As of:



b. Planned Number CSUs to Complete Detailed Design: Data is obtained from the estimated number of CSUs each reporting period that will complete detailed design and be documented in the SDD and IDD.

c. Actual Number of Requirements Allocated: Data is obtained from actual number of CSUs completing detailed design and documented in the SDD and IDD for each reporting period.

d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations.

4.1.3.5 An Example of the Indicator. An example of the Preliminary Design Status indicator is shown in Figure 4-3.

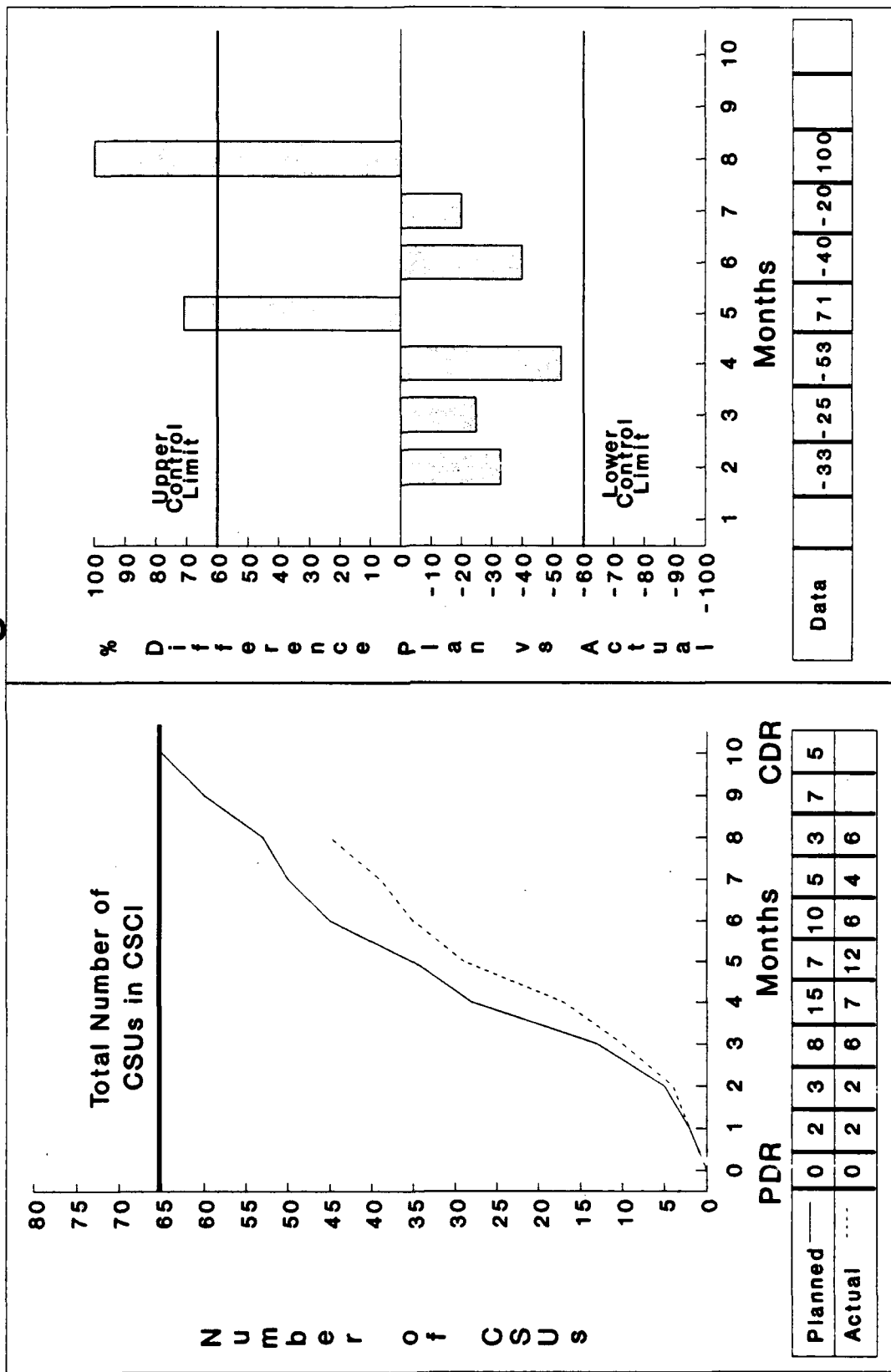
4.1.2.6 Using the Indicator. The indicator provides information concerning how well the detailed design activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the detailed design effort while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the detailed design task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.1.3.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Have separate indicators for the SDD and IDD
- b. Breakout Planned and Actual data for specific sections of the SDD and IDD (Sections 3 and 4).
- c. Change frequency of updates to meet your needs.
- d. Change time period of indicator to meet your needs.

Figure 4-3 Example of Detailed Design Status Indicator

# CSCI (Name of CSCI) Detailed Design Status



As of:

#### 4.1.4 Code and Unit Test Status

4.1.4.1 Purpose. The purpose of the Code and Unit Test Status indicator is to track the progress of coding and testing the CSUs.

4.1.4.2 Frequency of Update. This indicator will be updated monthly from the Critical Design Review until the Test Readiness Review has been successfully completed.

4.1.4.3 Update Criteria. Data for updates will be obtained by weekly inspections to determine the number of CSUs that have been coded AND successfully passed testing.

4.1.4.4 Indicator Inputs. The indicator is constructed from the following information:

a. Total Number of CSUs in the CSCI: Data is obtained from estimates or actual numbers if known.

b. Planned Number of CSUs to Complete Code and Testing: Data is obtained from the estimated number of CSUs each reporting period that will complete coding and testing.

c. Actual Number of CSUs to Complete Code and: Data is obtained from actual number of CSUs completing coding and testing for each reporting period.

d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

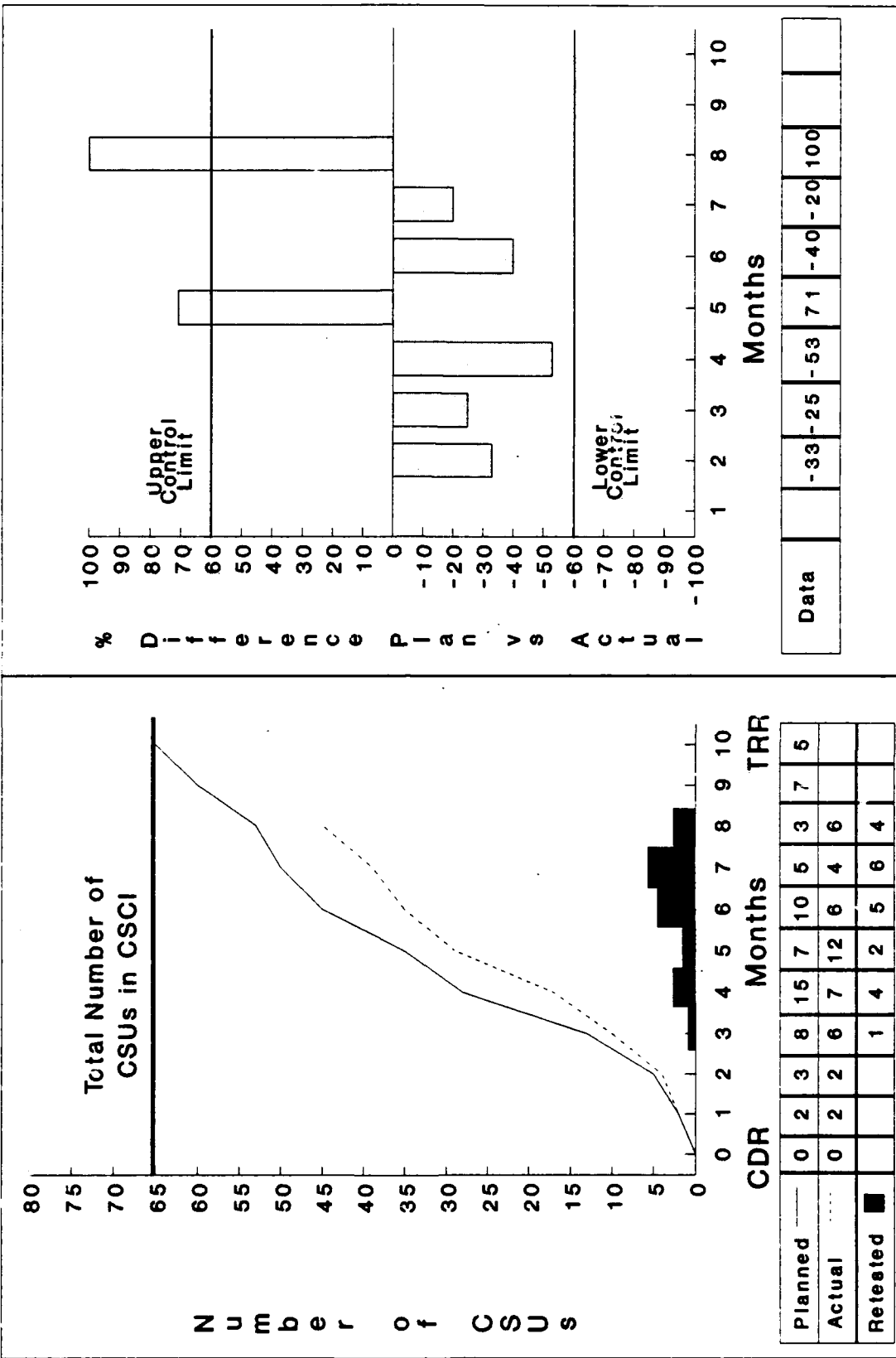
e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations.

4.1.4.5 An Example of the Indicator. An example of the Code and Unit Test Status indicator is shown in Figure 4-4.

4.1.4.6 Using the Indicator. The indicator provides information concerning how well the coding and testing activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the coding and testing effort while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the coding and testing task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual,

Figure 4-4 Example of Code and Unit Test Status Indicator

# CSCI (Name of CSCI) Code and Unit Test Status

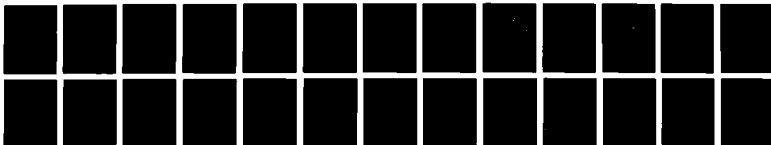


AD-7828 424

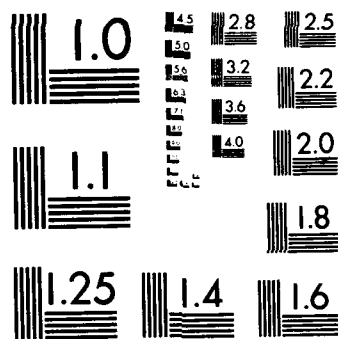
DEVELOPMENT OF A STANDARD SET OF SOFTWARE REQUIREMENTS  
FOR AERONAUTICAL SYSTEMS CENTER(U) AIR FORCE INST OF  
TECH WRIGHT-PATTERSON AFB OH SCHOOL OF SYSTEMS AND  
LOGISTIC B J AYERS ET AL. SEP 92

UNCLASSIFIED

NL



END  
FILMED  
DTIC



RESOLUTION TEST CHART  
1010-10A

this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.1.4.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Change time period of indicator to meet your needs.

#### 4.1.5 Integration Status

4.1.5.1 Purpose. The purpose of the Integration Status indicator is to track the progress of integrating the CSUs into the CSCI.

4.1.5.2 Frequency of Update. This indicator will be updated monthly from the Test Readiness Review until integration has been successfully completed.

4.1.5.3 Update Criteria. Data for updates will be obtained by weekly inspections to determine the number of CSUs that have been successfully integrated into a CSCI. This requires the integration of a group of CSUs, usually comprising a CSC, and passing testing requirements for the performance of the CSCI.

4.1.5.4 Indicator Inputs. The indicator is constructed from the following information:

a. Total Number of CSUs in the CSCI: Data is obtained from estimates or actual numbers if known.

b. Planned Number of CSUs to Complete Integration: Data is obtained from the estimated number of CSUs each reporting period that will complete integration.

c. Actual Number of CSUs to Complete Integration: Data is obtained from actual number of CSUs completing integration for each reporting period.

d. Number of CSUs Undergoing Retesting: Data is obtained from the actual number of CSUs that have failed integration, been recoded, and are being retested.

e. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

f. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations.

4.1.5.5 An Example of the Indicator. An example of the Integration Status indicator is shown in Figure 4-5.

4.1.5.6 Using the Indicator. The indicator provides information concerning how well the integration activities are meeting the estimate or plan. The top graph of the indicator provides an overall status of the integration effort while the bottom graph indicates how well the process is performing. The top graph also shows how many CSUs underwent retesting for each reporting period. Even though the actuals may be close to the planned, a significant portion of the actuals may be retests and not new CSUs. As indicated by this example, the top graph indicates that the integration task may be getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.1.5.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Change time period of indicator to meet your needs.
- c. Add information concerning number of CSUs recoded or modified.

## 4.2 Cost

There are two indicators associated with the cost area: man months of effort, and software size. The man months of effort indicator provides the basis for making cost estimates. The software size indicator provides a basis for determining and checking inputs to software cost models.



# CSCI (Name of CSCI) Integration Status

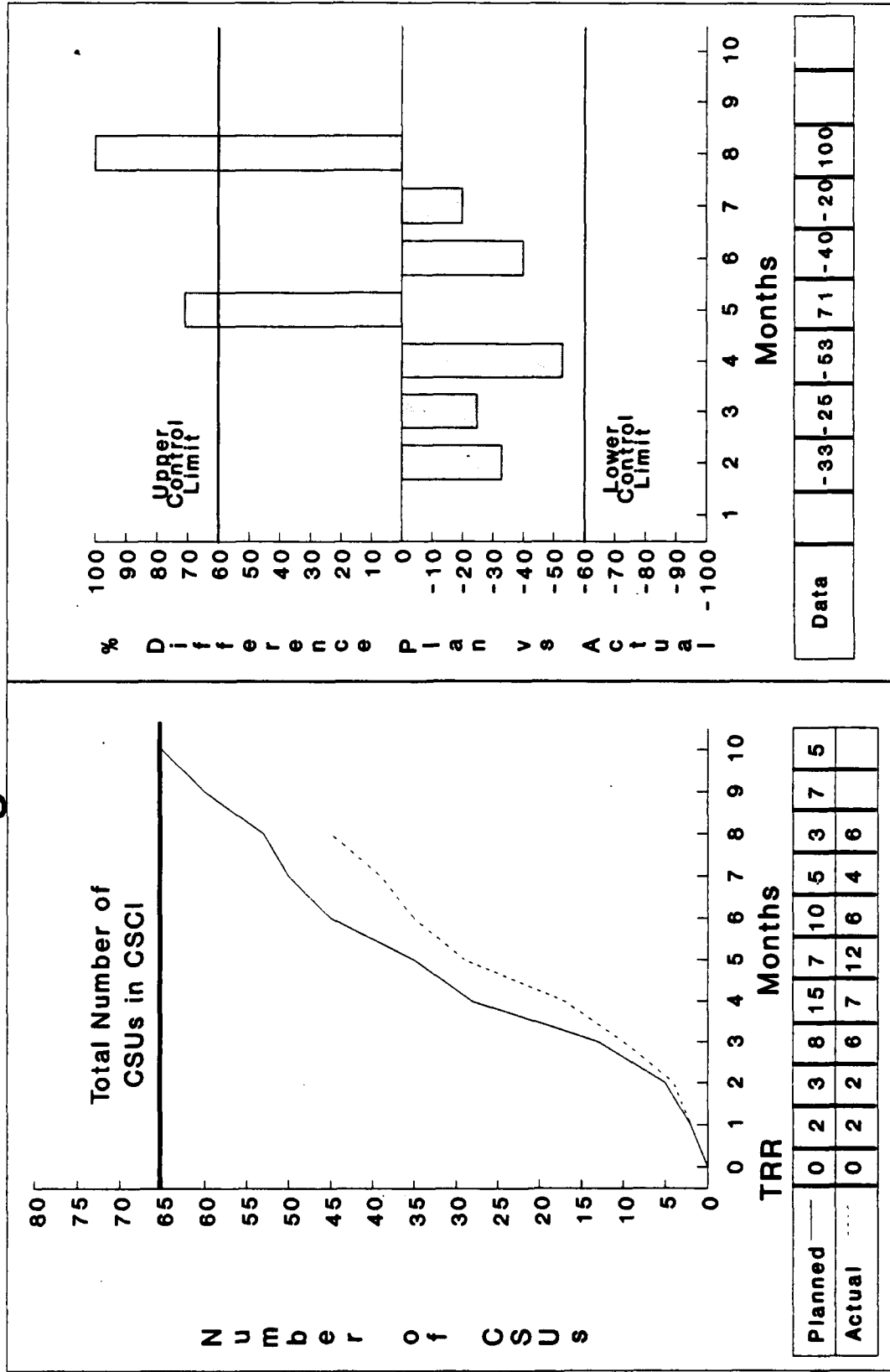


Figure 4-5 Example of Integration Status Indicator

#### 4.2.1 Man Months of Effort

4.2.1.1 Purpose. The purpose of the Man Months of Effort indicator is to track the level of effort being applied to the software development effort. This information can be used to indicate problems that need corrective actions and provide a basis for improving the inputs to software cost models.

4.2.1.2 Frequency of Update. This indicator will be updated monthly from contract award until completion of the program.

4.2.1.3 Update Criteria. Data for updates will be obtained from the cost/schedule control systems or internal timekeeping of the software developer.

4.2.1.4 Indicator Inputs. The indicator is constructed from the following information:

a. Total Number of Man Months of Effort: Data is obtained from estimates, models, proposals, etc.

b. Planned Number of Man Months: Data is obtained from the estimated number of man months to be expended on the program each reporting period.

c. Actual Number of Man Months: Data is obtained from actual number of man months of effort being expended for each reporting period.

d. Percent Difference Between Planned and Actual: Data is obtained from the following equation:

$$\frac{\text{Actual} - \text{Planned}}{\text{Planned}} \times 100$$

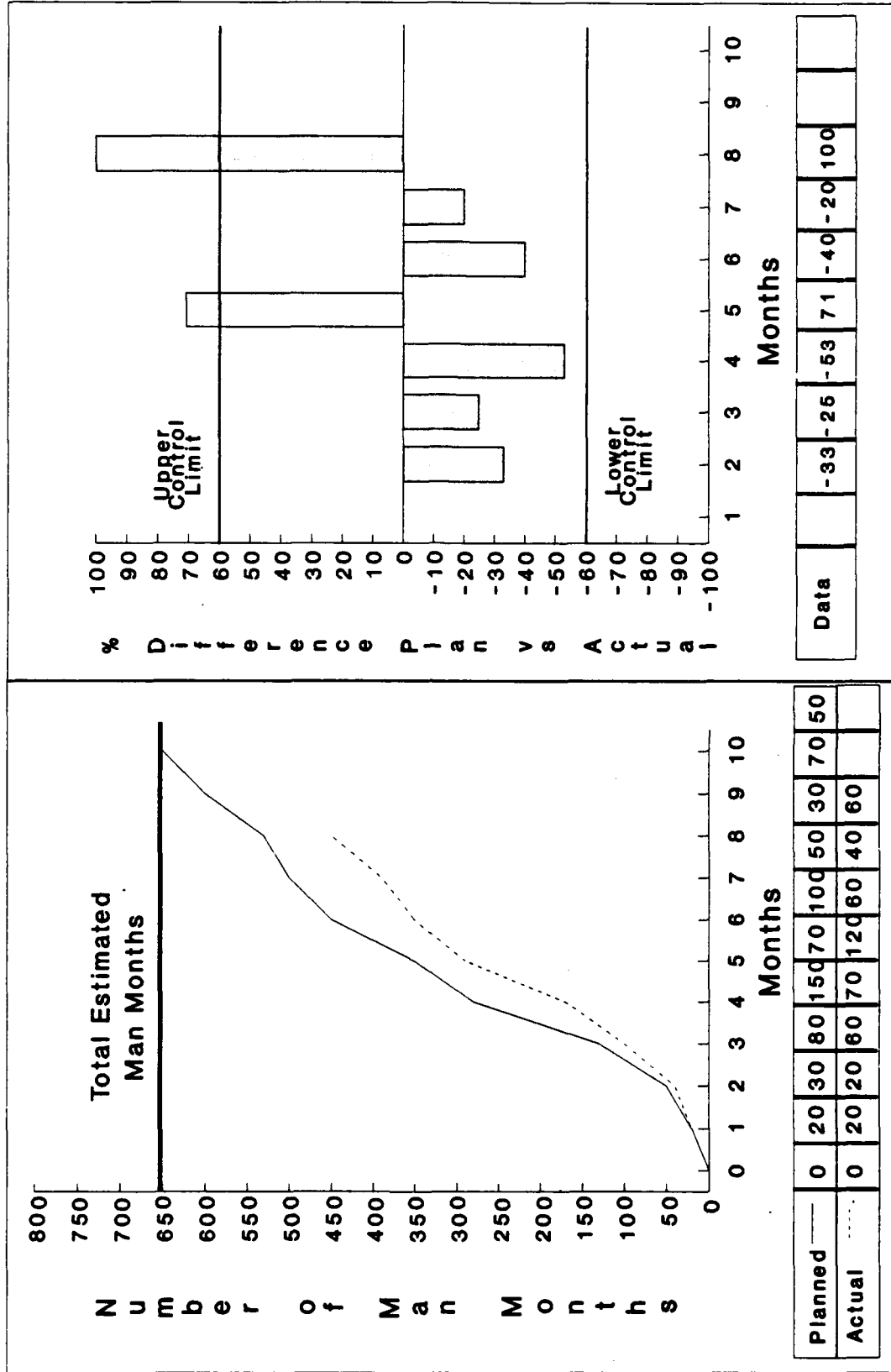
e. Upper and Lower Control Limits. Limits are set by taking the mean of the data points and calculating the standard deviation for the sample. Typically the control limits are set at  $\pm$  two standard deviations.

4.2.1.5 An Example of the Indicator. An example of the Man Months of Effort indicator is shown in Figure 4-6.

4.2.1.6 Using the Indicator. The indicator provides information concerning how well the actual man months match the estimated or predicted man months. In addition, it also provides information to be used to improve future estimates of levels of effort. The top graph of the indicator provides an overall status of how well the actual matches the estimate while the bottom graph indicates how well the process is performing. As indicated by this example, the top graph indicates that the integration task may be

Figure 4-6 Example of Man Months of Effort Indicator

# CSCI (Name of CSCI) Man Months of Effort



As of:

getting back on track, but the bottom graph shows large deviations between the planned and actual activities. When there is a large deviation between the planned and actual, this should prompt immediate management attention to identify the cause of the variation and try to eliminate it. The elimination of variability in the process will bring the difference between the planned and actual closer together.

4.2.1.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Change time period of indicator to meet your needs or correspond to outputs of cost models.
- c. If Cost/Schedule Status Reports or Cost Performance Reports can provide the same data, then this indicator may be deleted. The key is to provide sufficient visibility of software within the Work Breakdown Structure to ensure the adequate information is obtained.

#### 4.2.2 Software Size

4.2.2.1 Purpose. The purpose of the Software Size indicator is to track the size of the software being developed. This information can be used to provide a basis for improving the inputs to software cost models.

4.2.2.2 Frequency of Update. This indicator will be updated monthly from contract award until completion of the program.

4.2.2.3 Update Criteria. Data for updates will be obtained from inspections of code that has been written.

4.2.2.4 Indicator Inputs. The indicator is constructed from the following information:

- a. Original Estimate of Total KSLOC: Data is obtained from the original estimate, model output, proposals, etc., that was derived at the beginning of the program prior to contract award.
- b. Original Estimate of New KSLOC: Data is obtained from the original estimate, model output, proposals, etc., that was derived at the beginning of the program prior to contract award.
- c. Original Estimate of Reusable KSLOC: Data is obtained from the original estimate, model output, proposals, etc., that was derived at the beginning of the program prior to contract award.

d. Original Estimate of Modified KSLOC: Data is obtained from the original estimate, model output, proposals, etc., that was derived at the beginning of the program prior to contract award.

e. Current Estimate of Total KSLOC: Data is obtained from the current estimate of how many KSLOC are expected at the completion of the program.

f. Planned Number of Total KSLOC to be Developed: Data is obtained from cost models, expert judgment, etc.

g. Actual Number of Total KSLOC Developed: Data is obtained from actual total number of KSLOC developed to date.

h. Actual Number of New KSLOC Developed: Data is obtained from actual number of new KSLOC developed to date.

i. Actual Number of Reused KSLOC Developed: Data is obtained from actual number of reused KSLOC to date.

j. Actual Number of Modified KSLOC Developed: Data is obtained from actual number of modified KSLOC to date.

4.2.2.5 An Example of the Indicator. An example of the Software Size indicator is shown in Figure 4-7.

4.2.2.6 Using the Indicator. The indicator provides information comparing the original, current, and actual estimates of the software size. By comparing the actual to the original estimate, the accuracy of the original estimate can be determined. By comparing the actual to the current estimate gives insight into what is going on in the program and if there is need for management attention. The additional information concerning the breakout of the software between new, reused, and modified allows for comparing against the original inputs that may have been derived from cost models, and it provides data to be used to improve the inputs to and estimates from cost models in the future.

4.2.2.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

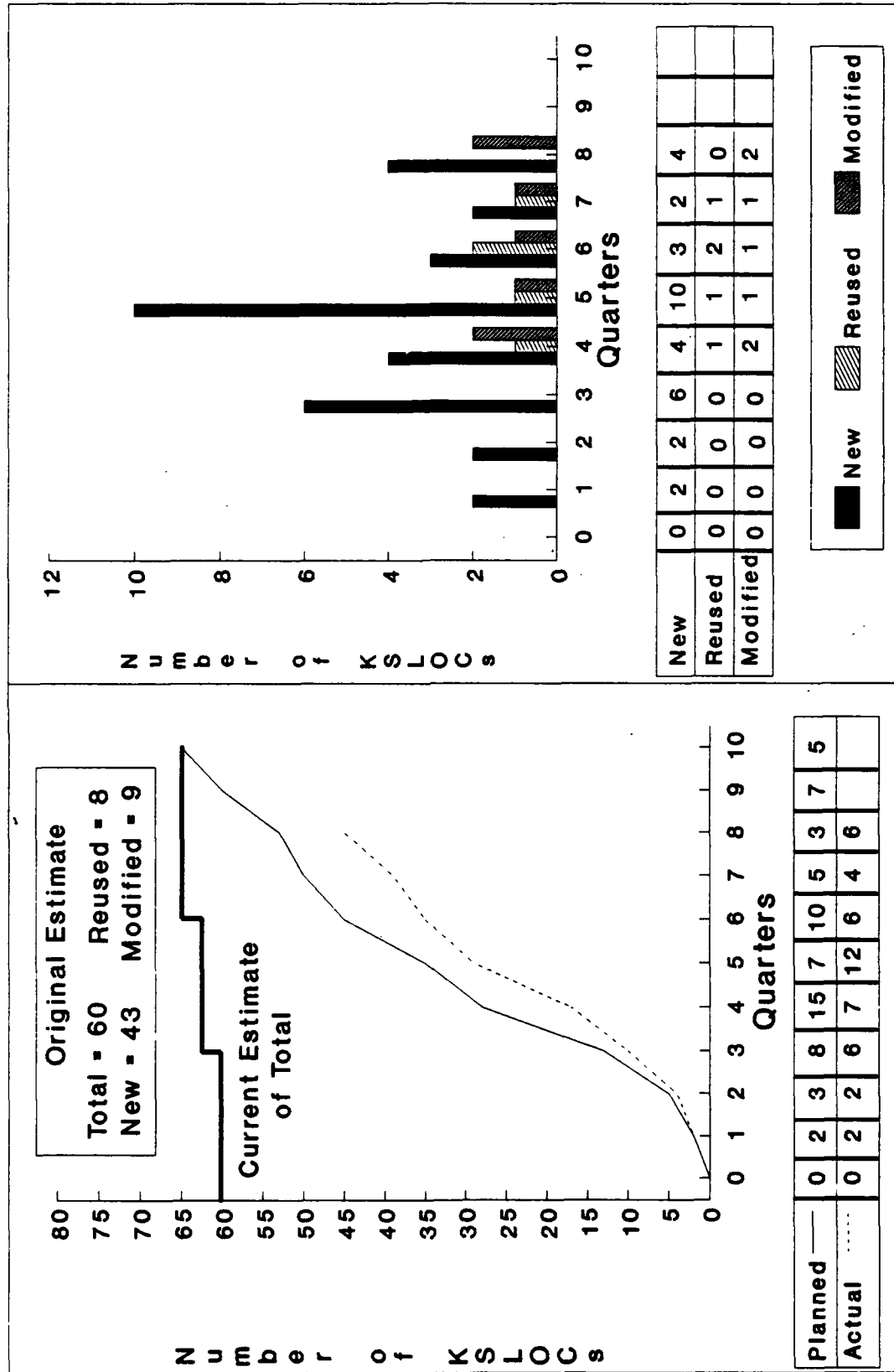
a. Change frequency of updates to meet your needs.

b. Change time period of indicator to meet your needs or correspond to outputs of cost models.

c. Eliminate the detailed breakout of the size.

d. Each language, i.e. Ada, assembler, should have its own indicator to report factual information.

# CSCI (Name of CSCI) Software Size



As of:

Figure 4-7 Example of Software Size Indicator

### 4.3 Requirements

There are two indicators associated with the requirements area: Requirements Stability and Design Stability

#### 4.3.1 Requirements Stability

4.3.1.1 Purpose. The purpose of the Requirements Stability indicator is to track the number of changes that are added to, deleted from, or modified within the SRS and IRS for a CSCI. Because unstable requirements can increase schedule and cost, and requirements reflect the needs of the users, this information can be used to indicate problems that need corrective actions.

4.3.1.2 Frequency of Update. This indicator will be updated monthly from the time the SRS and IRS are baselined until completion of the program.

4.3.1.3 Update Criteria. Whenever a proposed change to the SRS or IRS is formally approved, the number and type of requirements changes will provide the data for updates.

4.3.1.4 Indicator Inputs. The indicator is constructed from the following information:

a. Total Number of Requirements in the CSCI: Data is obtained from the requirements in the SRS and IRS plus any added requirements for the reporting period minus any deleted requirements for the reporting period.

b. Total Number of Changes: Data is obtained from the number of formally approved changes made to the SRS and IRS since they were baselined.

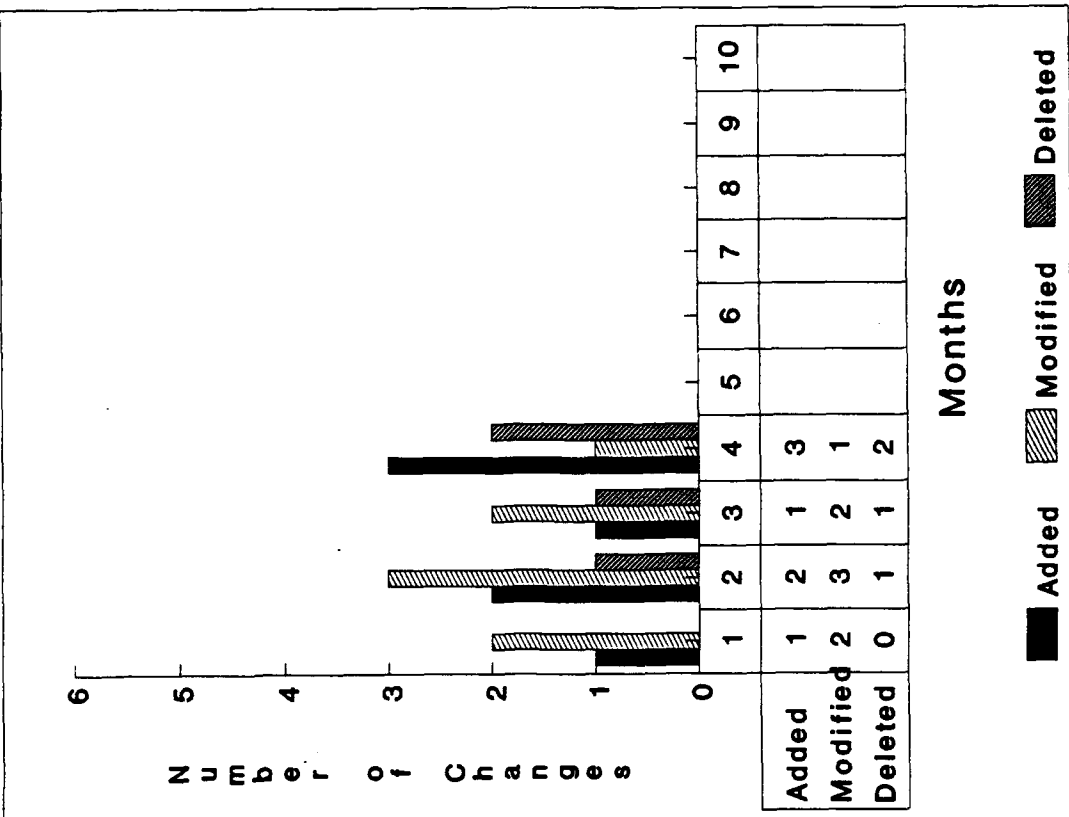
c. Number of Added Requirements: Data is obtained from formally approved changes to the SRS and IRS for that reporting period.

d. Number of Deleted Requirements: Data is obtained from formally approved changes to the SRS and IRS for that reporting period.

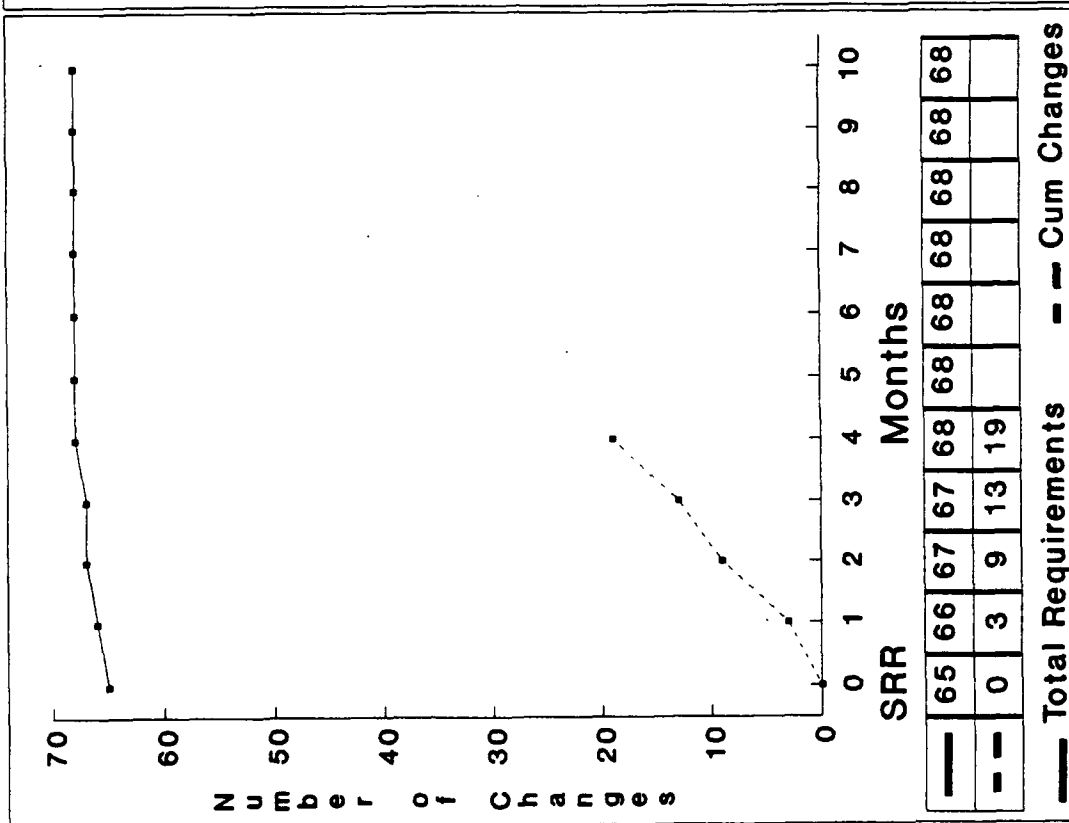
e. Number of Modified Requirements: Data is obtained from formally approved changes to the SRS and IRS for that reporting period.

4.3.1.5 An Example of the Indicator. An example of the Requirements Stability indicator is shown in Figure 4-8.

# CSCI (Name of CSCI) Requirements Stability



**As of:**



**Figure 4-8 Example of Requirements Stability Indicator**



4.3.1.6 Using the Indicator. The indicator provides information concerning how stable the requirements are for the CSCI. An increasing number of changes can indicate unstable requirements that can increase schedule and cost. The breakout of the number of changes can provide increased visibility into which types of requirements are dominating the amount of changes being made.

4.3.1.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Do not breakout the types of changes.

#### 4.3.2 Design Stability

4.3.2.1 Purpose. The purpose of the Design Stability indicator is to track the number of changes that are added to, deleted from, or modified within the SDD and IDD for a CSCI. Because changes to the design can increase schedule and cost this information can be used to indicate problems that need corrective actions.

4.3.2.2 Frequency of Update. This indicator will be updated monthly from the time the SDD and IDD are baselined until completion of the program.

4.3.2.3 Update Criteria. Whenever a proposed change to the SDD or IDD is formally approved, the number and type of requirements changes will provide the data for updates.

4.3.2.4 Indicator Inputs. The indicator is constructed from the following information:

- a. Total Number of Designs in the CSCI: Data is obtained from the designs in the SDD and IDD plus any added designs for the reporting period minus any deleted designs for the reporting period.
- b. Total Number of Changes: Data is obtained from the number of formally approved changes made to the SDD and IDD since they were baselined.
- c. Number of Added Requirements: Data is obtained from formally approved changes to the SDD and IDD for that reporting period.
- d. Number of Deleted Requirements: Data is obtained from formally approved changes to the SDD and IDD for that reporting period.

e. Number of Modified Requirements: Data is obtained from formally approved changes to the SDD and IDD for that reporting period.

4.3.2.5 An Example of the Indicator. An example of the Design Stability indicator is shown in Figure 4-9.

4.3.2.6 Using the Indicator. The indicator provides information concerning how stable the designs are for the CSCI. An increasing number of changes can indicate unstable designs that can increase schedule and cost. The breakout of the number of changes can provide increased visibility into which types of requirements are dominating the amount of changes being made.

4.3.2.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Do not breakout the types of changes.

#### 4.4 Software Performance

There are three indicators associated with the software performance area: I/O Bus Throughput Capability, Processor Memory Utilization, Processor Throughput Utilization.

##### 4.4.1 I/O Bus Throughput Capability

4.4.1.1 Purpose. The purpose of the I/O Bus Throughput Capability indicator is to track the estimated and actual throughput of the Bus being analyzed.

4.4.1.2 Frequency of Update. This indicator will be updated bi-monthly from the Software Specification Review until completion of the program.

4.4.1.3 Update Criteria. Estimates will be updated with actual data measured from the target hardware.

4.4.1.4 Indicator Inputs. The indicator is constructed from the following information:

- a. Maximum Capability of the Bus: The maximum capability of the Bus measured in KBPS and percentage.
- b. Effective Capability of the Bus: The effective capability of the Bus measured in KBPS and as a percentage of the Maximum.

# CSCI (Name of CSCI) Design Stability

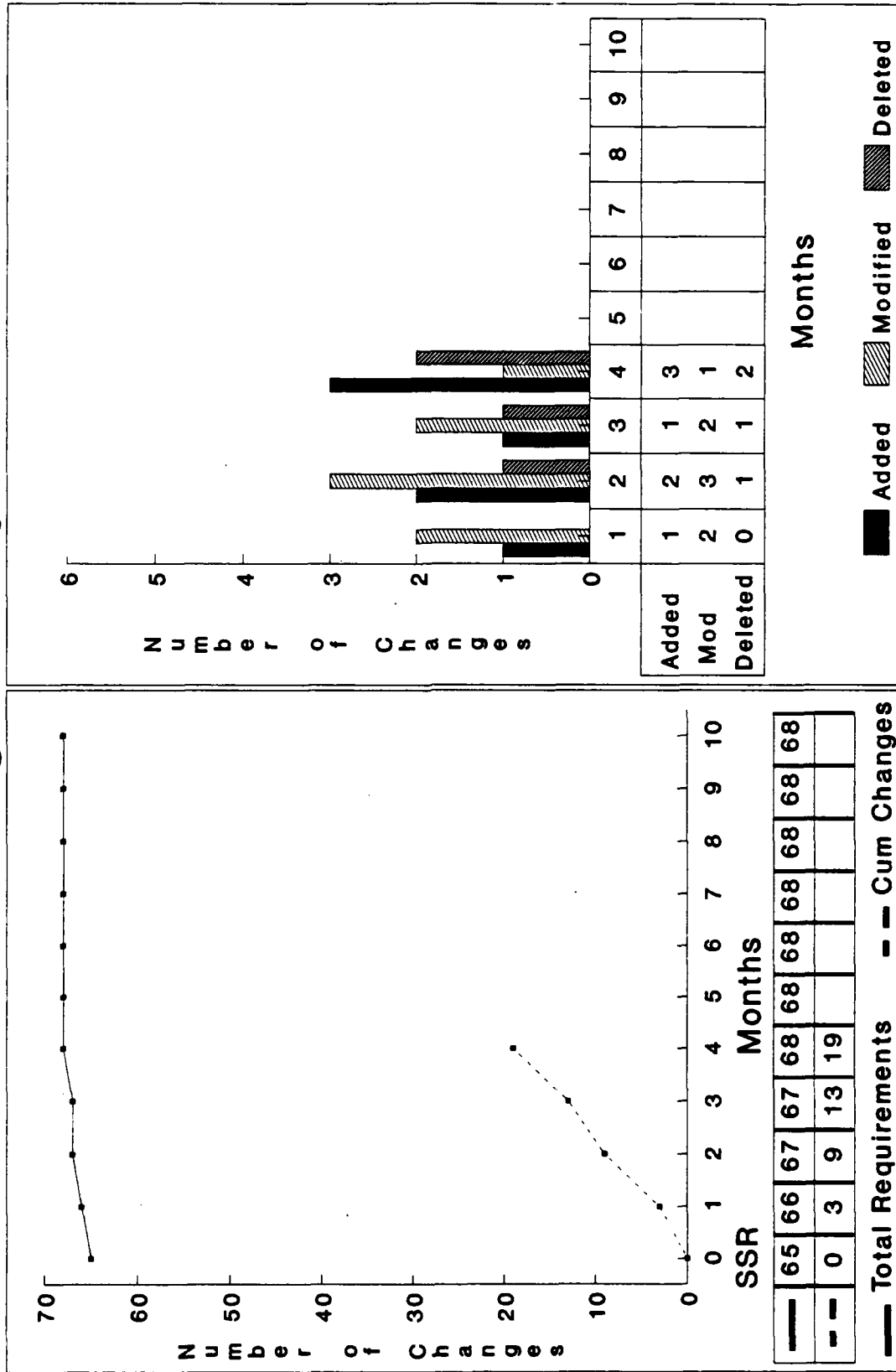


Figure 4-9 Example of Design Stability Indicator

As of:

c. Management Reserve: The management reserve of the Bus throughput.

d. Estimated Bus Throughput: The estimated throughput of the Bus.

e. Actual Bus Throughput: The actual measured throughput of the BUS.

4.3.1.5 An Example of the Indicator. An example of the I/O Bus Throughput Capability indicator is shown in Figure 4-10.

4.3.1.6 Using the Indicator. The indicator provides information concerning the throughput of the Bus. It indicates when management reserve has been used up and the Bus can no longer efficiently handle the workload of the software. This allows for corrective actions to be generated to resolve the problem.

4.3.1.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program: Change frequency of updates to meet your needs.

#### 4.4.2 Processor Memory Utilization

4.4.2.1 Purpose. The purpose of the Processor Memory Utilization indicator is to track the estimated and actual memory usage for all the software allocated to that processor.

4.4.2.2 Frequency of Update. This indicator will be updated bi-monthly from the Software Specification Review until completion of the program.

4.4.2.3 Update Criteria. The original estimates will be updated with actual software that has been generated, modified, or reused.

4.4.2.4 Indicator Inputs. The indicator is constructed from the following information:

a. Maximum Capacity of the Memory: The maximum capacity of the memory measured in KSLOC and percentage.

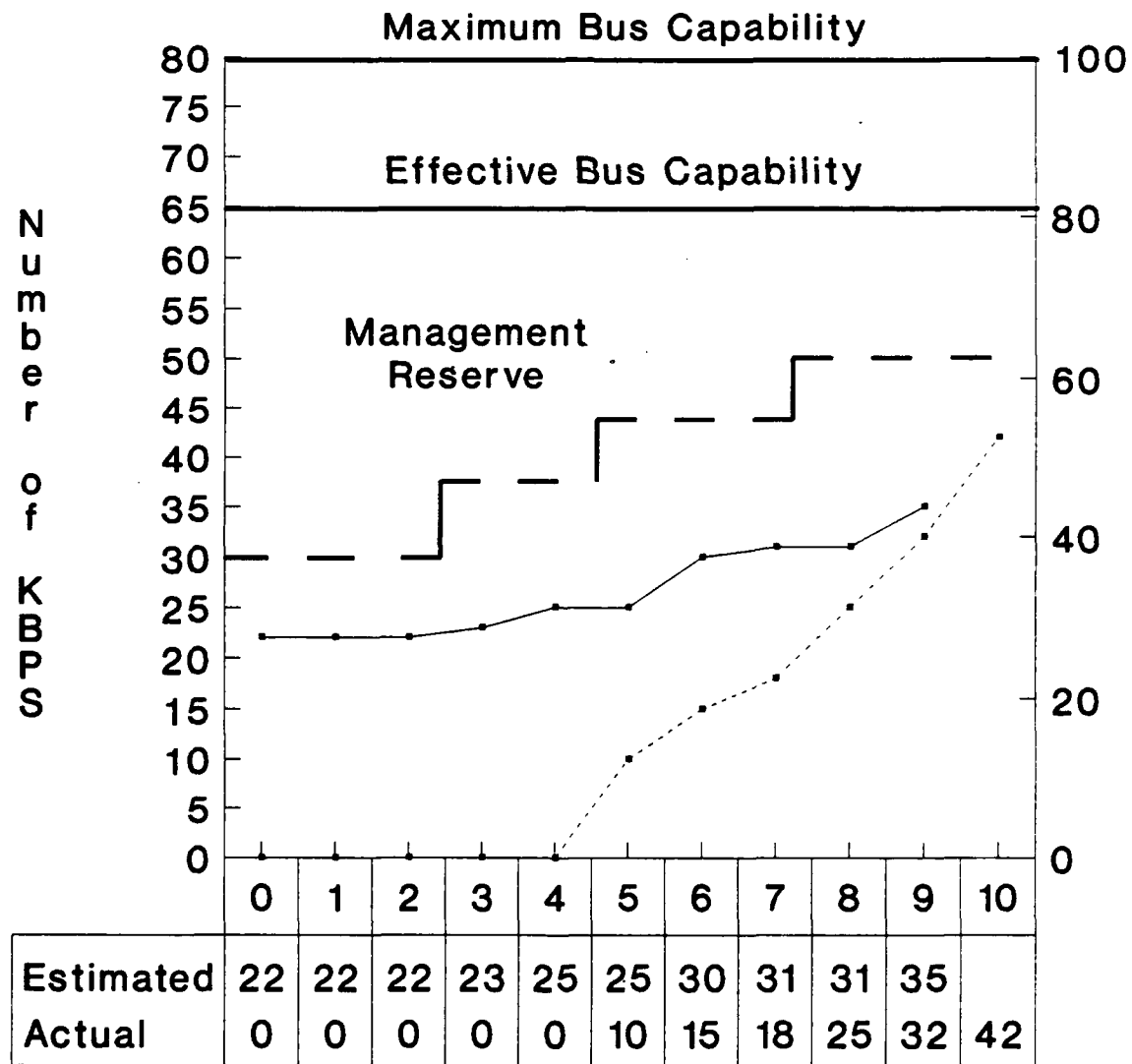
b. Effective Capacity of the Bus: The effective capacity of the memory measured in KSLOC and as a percentage of the Maximum.

c. Management Reserve: The management reserve of the memory.

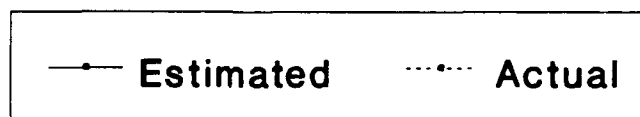
d. Estimated Memory Utilization: The estimated memory utilization of the processor.

Figure 4-10 Example of I/O Bus Throughput Capability Indicator

## I/O Bus (Name of Bus) Throughput Capability



Quarters



e. Actual Memory Utilization: The actual memory utilization of the processor.

4.4.2.5 An Example of the Indicator. An example of the Processor Memory Utilization indicator is shown in Figure 4-11.

4.4.2.6 Using the Indicator. The indicator provides information concerning the memory utilization of the Bus. It indicates when the management reserve has been used up and the processor memory can no longer handle the software. This allows for corrective actions, like rebalancing the resources, to be generated to resolve the problem.

4.4.2.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Breakout the memory utilization for separate CSCIs that all are allocated to the same processor.

#### 4.4.3 Processor Throughput Utilization

4.4.2.1 Purpose. The purpose of the Processor Throughput Utilization indicator is to track the estimated and actual execution time for all the software allocated to that processor.

4.4.2.2 Frequency of Update. This indicator will be updated bi-monthly from the Software Specification Review until completion of the program.

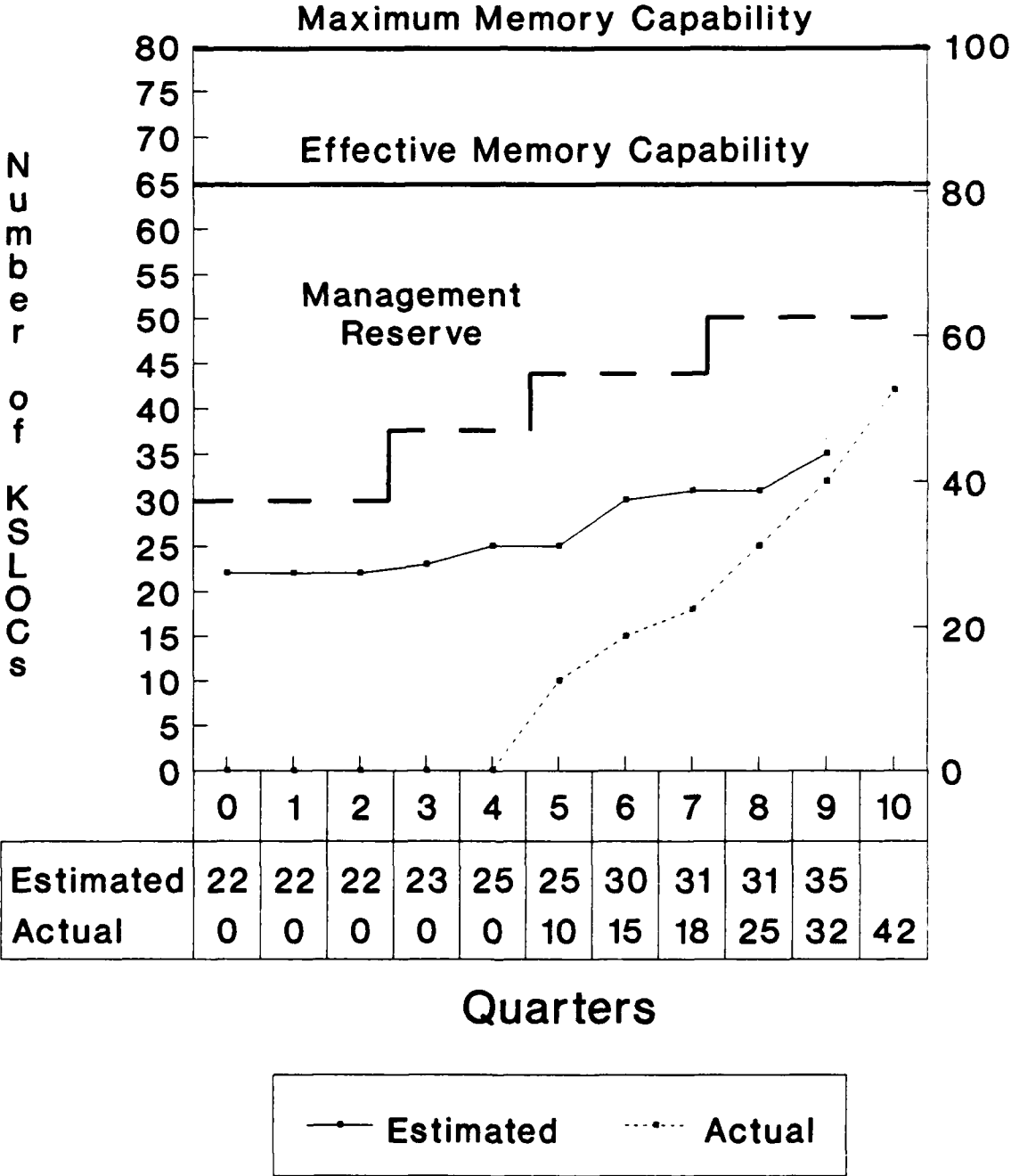
4.4.2.3 Update Criteria. The original estimates will be updated with actual measurements taken from the target hardware for software that has been generated, modified, or reused.

4.4.2.4 Indicator Inputs. The indicator is constructed from the following information:

- a. Maximum Processor Execution Capability: The maximum execution capability of the processor measured in MIPS and percentage.
- b. Effective Processor Execution Capability: The effective execution capability of the processor measured in MIPS and as a percentage of the Maximum.
- c. Management Reserve: The computational management reserve of the processor.

Figure 4-11 Example of Processor Memory Utilization Indicator

# Processor (Name of Processor) Memory Utilization



d. Estimated Processor Computational Requirements: The estimated computational requirements for the processor.

e. Actual Memory Utilization: The actual measured throughput of the processor.

4.4.2.5 An Example of the Indicator. An example of the Processor Throughput Utilization indicator is shown in Figure 4-12.

4.4.2.6 Using the Indicator. The indicator provides information concerning the throughput of the processor. It indicates when the management reserve has been used up and the processor can no longer handle the software. This allows for corrective actions, like rebalancing the resources, to be generated to resolve the problem.

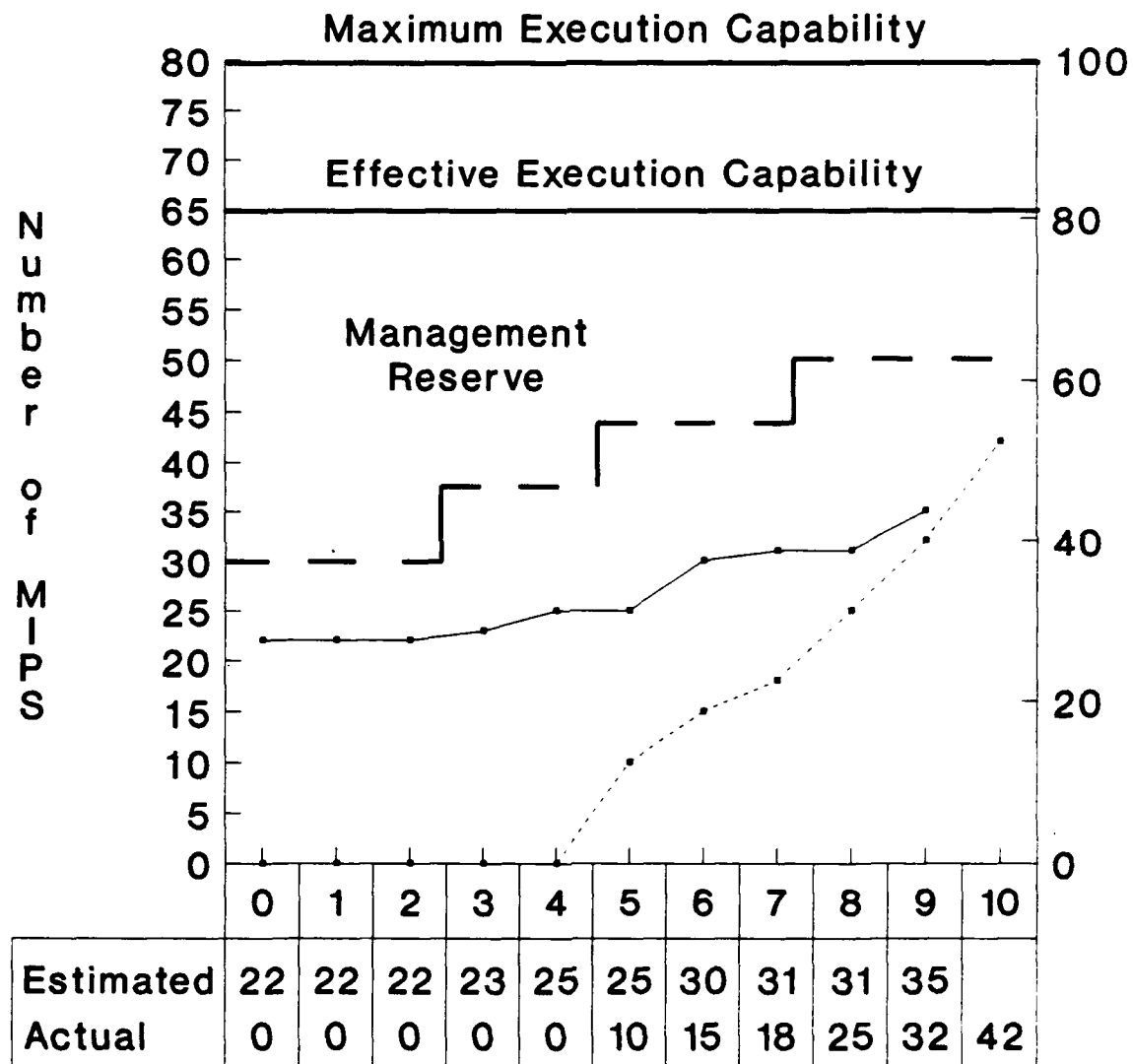
4.4.2.7 Tailoring Suggestions. The following suggestions may be utilized to tailor the indicator to a specific program:

- a. Change frequency of updates to meet your needs.
- b. Breakout the memory utilization for separate CSCIs that all are allocated to the same processor.

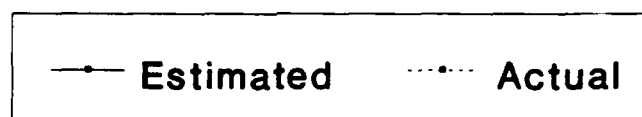


Figure 4-12 Example of Processor Throughput Utilization Indicator

## Processor (Name of Processor) Throughput Utilization



Quarters



### Section 5.0 Other Sources of Indicator Information

1. AFSCP 800-43, Software Management Indicators
2. AFSCP 800-14, Software Quality Indicators
3. ESD-TR-88-01, Software Management Indicators
4. SEI-CM-12-1.1, Software Metrics
5. IEEE Standard 982.1, IEEE Standard Dictionary of Measures to Produce Reliable Software
6. ASDP 700-8, ASD Metrics Handbook

### Bibliography

1. Aeronautical Systems Division. ASD Metrics Handbook. ASDP 700-8. Wright-Patterson Air Force Base OH: HQ ASD, 30 April 1992.
2. Air Force Systems Command. Software Management Indicators. AFSCP 800-43. Andrews Air Force Base DC: HQ AFSC, 31 August 1990.
3. Air Force Systems Command. Software Quality Indicators. AFSCP 800-14. Andrews Air Force Base DC: HQ AFSC, 20 January 1987.
4. Air Force Institute of Technology. Introduction to Acquisition Management. Wright-Patterson AFB OH, April 1992.
5. ASD/ENASC. "AFIT Call for MS Thesis Topics in Computer Engineering." Memorandum For Record, 28 May 91.
6. Basili, Victor R, and David M. Weiss. "A Methodology for Collecting Valid Software Engineering Data," IEEE Transactions on Software Engineering, SE-10: 729-732 (November 1984).
7. Booch, Grady. Software Engineering with Ada (Second Edition). Menlo Park CA: The Benjamin/Cummings Publishing Company, 1987.
8. Cho, Chin-Kuei. Quality Programming. New York NY: John Wiley & Sons, 1987.
9. Conte, Samuel D. et al. Software Engineering Metrics and Models. Menlo Park CA: The Benjamin/Cummings Publishing Company, 1986.
10. Defense Systems Management College. Program Manager's Handbook. 1989.
11. DeMarco, Tom. Controlling Software Projects. Engelwood Cliffs NJ: Prentice-Hall Inc., 1982.
12. Deming, W. Edwards. Out of the Crisis. Cambridge MA: Massachusetts Institute of Technology, 1986.
13. Department of Defense. Defense Acquisition Program Procedures. DOD Directive 5000.2. Washington: Government Printing Office, 23 February 1991.
14. Emory, William C. Business Research Methods. Homewood IL: Richard D. Irwin, Inc, 1991.
15. General Electric Company. Software Engineering Handbook. New York NY: McGraw- Hill Co., 1986.
16. Gilb, Tom. Software Metrics. Cambridge MA: Winthrop Publishers, Inc., 1977.
17. Grady, Robert B. and Deborah L. Caswell. Software Metrics: Establishing a Company-Wide Program. Engelwood Cliffs NJ: Prentice-Hall Inc., 1987.

18. Humphrey, Watts S. Managing the Software Process. Reading MA: Addison-Wesley Publishing Company, 1990.
19. Institute of Electrical and Electronics Engineers. IEEE Standard Dictionary of Measures to Produce Reliable Software. IEEE Std 982.1. New York NY: Institute of Electrical and Electronics Engineers, 1989.
20. Jones, Capers. Programming Productivity. New York NY: McGraw- Hill Co., 1986.
21. Leenders, Michiel R. Purchasing and Materials Management (Eighth Edition). Homewood IL: Richard D Irwin, Inc, 1985.
22. Marciniak, John and Donald Reifer. Software Acquisition Management. New York NY: John Wiley & Sons, 1990.
23. Marsh, Alton. "Pentagon Up Against a Software Wall," Government Executive, 22: 62-63 (May 1990).
24. Miles, Vernon H. et al. Adapting Software Development Policies to Modern Technology. Washington DC: National Academy Press, 1989.
25. Moen, Ronald D. et al. Improving Quality Through Planned Experimentation. New York NY: McGraw-Hill Book Company, 1991.
26. Neave, Henry R. The Deming Dimension. Knoxville TN: SPC Press Inc., 1990.
27. Pressman, Roger S. Software Engineering: A Practitioner's Approach (Second Edition). New York NY: McGraw-Hill Book Company, 1987.
28. Scherkenbach, William. Continuous Improvement. Knoxville TN: SPC Press Inc., 1991.
29. Software Engineering Institute. Software Metrics. SEI-CM-12-1.1. Pittsburgh PA: Carnegie-Mellon University, December 1988.
30. The MITRE Corporation. Software Management Metrics. Report ESD-TR-88-001. Bedford MA. The MITRE Corporation, 1988.
31. Walton, Mary. Deming Management at Work. New York NY: G. P. Putnam's Sons, 1990.
32. Walton, Mary. The Deming Management Method. New York NY: Putnam Publishing Group, 1986.

### Vita

Captain Bradley J. Ayres was born on 19 August 1959 in St. Louis, Missouri. He graduated from Lafayette High School in Ellisville, Missouri in 1977 and attended the University of Missouri at Columbia, Missouri, graduating with a Bachelor of Science in Chemical Engineering in May 1982. Captain Ayres served his first tour of duty at Wright Patterson AFB as an Acquisition Project Officer in the Deputate for Reconnaissance, Strike, and Electronic Warfare Systems, ASD/RW, where he was the program manager for the Infrared Search and Track System. Captain Ayres was then assigned to Headquarters Air Force Communications Command, Directorate for Acquisition and Integration, Scott AFB, Illinois. There he was the action officer responsible for oversight of the System Modernization program until entering the School of Systems and Logistics, Air Force Institute of Technology, in May 1991.

Permanent Address:

807 Crestland  
Ballwin, MO 63011

### Vita

Captain William M. Rock was born on 11 January 1960 in Hammond, Indiana. He graduated from Bremen High School in Midlothian, Illinois in 1978 and attended Illinois Institute of Technology, graduating with a Bachelor of Science in Electrical Engineering. He was commissioned in the Air Force in 1982, and has served undistinguished tours of duty at Tinker AFB, Langley AFB, and at Ft Huachuca. He has served in combat communications, engineering-installation, and test and evaluation of tactical communications in addition to a tour at Headquarters, Tactical Air Command. He entered the School of Systems and Logistics in May 1991 after his unit at Ft Huachuca was shut down.

**Permanent Address:**

1060 W. Addison

Chicago, Illinois 60611

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 1992	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE  DEVELOPMENT OF A STANDARD SET OF SOFTWARE INDICATORS FOR AERONAUTICAL SYSTEMS CENTER			5. FUNDING NUMBERS	
6. AUTHOR(S)  Bradley J. Ayres, Captain, USAF William M. Rock, Captain, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Air Force Institute of Technology, WPAFB OH 45433-6583			8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GSS/ENC/92D-1	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Mr Fred Banks ASC/ENASC WPAFB OH 45433			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT  Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)  This research effort was directed to the development of a standard set of software indicators to be used by ASC. The research was sponsored by ASC/ENASC, who indicated their desire for such a standard set of indicators to allow a database to be developed of program data that would be useful in making predictions about future software efforts. The research began with an attempt to characterize the software indicator environment at ASC, and also to perform a literature search for software indicators currently available. The desired result was an initial core set of indicators useful to software managers and engineers and to ENASC and also a methodology for the continuous refinement of the set of indicators as part of a process to better define the software indicator environment. Thirty-four interviews were done on software managers and engineers and the results showed some common areas of interest and some unique interests for managers and some for engineers.				
14. SUBJECT TERMS  Software, Software Management, Indicators, Metrics			15. NUMBER OF PAGES 409	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

## AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: AFIT/LSC, Wright-Patterson AFB OH 45433-9905.

1. Did this research contribute to a current research project?

a. Yes

b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

a. Yes

b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Please estimate what this research would have cost in terms of manpower and/or dollars if it had been accomplished under contract or if it had been done in-house.

Man Years \_\_\_\_\_

\$ \_\_\_\_\_

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3, above) what is your estimate of its significance?

a. Highly  
Significant

b. Significant

c. Slightly  
Significant

d. Of No  
Significance

5. Comments

\_\_\_\_\_  
Name and Grade

\_\_\_\_\_  
Organization

\_\_\_\_\_  
Position or Title

\_\_\_\_\_  
Address